Multi-camera Object Segmentation in Dynamically Textured Scenes Using Disparity Contours

Wei SUN

Doctor of Philosophy

Department of Electrical and Computer Engineering

McGill University

Montreal, Quebec

October 2006

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Doctor of Philosophy

Your file Votre référence
ISBN: 978-0-494-27844-4
Our file Notre référence
ISBN: 978-0-494-27844-4

DEDICATION

This thesis is dedicated to all my friends who have so generously and lovingly given me help and support during my difficult times.

ACKNOWLEDGEMENTS

I would like to acknowledge my supervisor, my thesis advisory committee, and my colleagues for their generous contributions to my thesis work. First, I would like to thank my supervisor, Jeremy R. Cooperstock, who provided me with an opportunity to work on this interesting project and essential research facilities in which to carry out my work.

For my work on camera calibration, my colleague Jianfeng Yin shared his own calibration experience with me, and referred me to a few important papers and websites on camera calibration. Don Pavlasek and Jozsef Boka of the Mechanics Workshop built the measurement assembly for 3D coordinate measurement.

For my development of the SRE image library, which performs real-time video acquisition, image format conversion, and X11 display on Linux system, my colleague, senior research associate Stephen P. Spackman helped me architect the library. In order to achieve synchronous video capture, my supervisor Jeremy R. Cooperstock showed me his sample multicast code.

For acquiring experimental data, my colleague Jianfeng Yin helped me with room geometry measurement and video capture. Jianfeng Yin, Yuwen Li and Zhi Qi also participated as subjects in the captured video sequences. Stephen P. Spackman helped me with computer system problems encountered during synchronous video acquisition. Our system administrator Jan Binder helped me with network setup and data storage.

For my work on object segmentation, the very first idea of falsifying the background assumption of a scene to extract the foreground objects arose from a discussion with Stephen P. Spackman. His insightful intuitions and valuable outsider's perspective inspired me throughout my thesis work.

I would like to thank my thesis advisory committee members, James J. Clark and Kaleem Siddiqi, for their constant encouragement and support during my study. I would like to show my appreciation to our system administrator Jan Binder, our secretary Cynthia Davidson, and our manager Marlene Gray for offering various help.

During my thesis writing, Stephen P. Spackman, James J. Clark and Jeremy R. Cooperstock all gave me helpful feedback. Finally, special thanks to my colleague Stephane Pelletier, who translated my thesis abstract from English to French.

ABSTRACT

This thesis presents a stereo-based object segmentation system that combines the simplicity and efficiency of the background subtraction approach with the capacity of dealing with dynamic lighting and background texture and large textureless regions. The method proposed here does not rely on full stereo reconstruction or empirical parameter tuning, but employs disparity-based hypothesis verification to separate multiple objects at different depths.

The proposed stereo-based segmentation system uses a pair of calibrated cameras with a small baseline and factors the segmentation problem into two stages: a well-understood offline stage and a novel online one. Based on the calibrated parameters, the offline stage models the 3D geometry of a background by constructing a complete disparity map. The online stage compares corresponding new frames synchronously captured by the two cameras according to the background disparity map in order to falsify the hypothesis that the scene contains only background. The resulting object boundary contours possess a number of useful features that can be exploited for object segmentation.

Three different approaches to contour extraction and object segmentation were experimented with and their advantages and limitations analyzed. The system demonstrates its ability to extract multiple objects from a complex scene with near real-time performance. The algorithm also has the potential of providing precise object boundaries rather than just bounding boxes, and is extensible to perform 2D and 3D object tracking and online background update.

ABRÉGÉ

Cette thèse présente un système de segmentation d'objets basé sur la vision stéréo incorporant la simplicité et l'efficacité de la soustraction de l'arrière-plan et pouvant fonctionner malgré des changements d'intensité lumineuse et la présence de grandes régions sans motifs dans l'arrière-plan. La méthode proposée ne dépend pas d'une reconstruction stéréo ou d'un ajustement empirique de paramètres, mais emploie la vérification d'hypothèses basées sur la disparité afin de séparer des objets situés à différentes profondeurs.

Le système de segmentation proposé utilise une paire de caméras calibrées ayant une ligne de base courte et divise le problème de segmentation en deux étapes: une étape autonome bien connue et une nouvelle étape en ligne. Suivant les paramètres de calibration, la première étape modélise la géométrie 3D de l'arrière-plan en construisant une carte de disparité. La seconde étape compare les images correspondantes acquises de manière synchronisée avec les deux caméras selon cette carte de disparité dans le but de contredire l'hypothèse d'une scène ne contenant que l'arrière-plan. Les contours d'objets résultants présentent certains attributs pouvant être exploités par la segmentation d'objets.

Trois approches distinctes pour l'extraction des contours et la segmentation d'objets ont été expérimentées et leurs avantages et limites ont été analysés. Le système a la capacité d'extraire plusieurs objets d'une scène complexe, et ce presque en temps réel. L'algorithme retourne aussi le contour précis des objets plutôt qu'une simple boîte de délimitation et peut être adapté pour faire du suivi d'objets en 2D et 3D et pour la mise à jour en temps réel de l'arrière-plan.

CONTRIBUTIONS

The work presented in this thesis makes contributions in two distinct areas of computer vision. The first set of contributions are with respect to the camera calibration work presented in Chapter 3.

- An empirical study of the impact of noise and training data quantity on calibration accuracy is conducted, using three publicly available techniques, through extensive experimentation with separate training and test data.
- A detailed comparison of various camera models is included to determine the relative importance of different distortion components.

The second set of contributions deal with the object segmentation work, which presents a multi-camera method to isolate and distinguish multiple foreground objects in a scene.

- It provides a solution in the presence of illumination and texture change in the background.
- The method generates not only the 2D locations of objects in images but also boundaries and depth information, without using full stereo reconstruction.
- It offers a new perspective on disparity analysis for depth extraction and 3D scene understanding.
- The approach suits generic object segmentation tasks as it does not require prior knowledge of the objects or impose any constraint on object shape.
- The system achieves near-real-time performance by avoiding slow online stereo matching, and is much faster than existing software based stereo segmentation systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AR	auto-regressive	29
BDM†	background disparity map	66
BHF†	background hypothesis falsification	66
CHMM	coupled hidden Markov model	30
DLT	direct linear transformation	38, 42
DSI	disparity-space image	19
EM	expectation maximization	29
FFT	fast Fourier transformation	29
FHV†	foreground hypothesis verification	99
GMM	Gaussian mixture model	25
HMC	hybrid Monte Carlo filter	32
HMM	hidden Markov model	26
MCMC	Markov chain Monte Carlo filter	32
MRF	Markov random field	33
NCC	normalized cross correlation	14
NCE	normalized calibration error	45
PCA	principal component analysis	23
RAC	radial alignment constraint	38, 41
SAD	sum of absolute differences	14, 67
SRE†	shared reality environment	52
SSD	sum of squared differences	14
VDM†	view difference map	67, 99

† Introduced in this thesis.

LIST OF NOTATIONS

$\top(\cdot)$, $\bot(\cdot)$	top or bottom end element	89
$\vdash(\cdot)$, $\dashv(\cdot)$	left or right border point	100
$\ltimes(\cdot)$, $\rtimes(\cdot)$	left or right neighbor	100, 103
BDM	background disparity map	67
C	contour image	75, 85
c	contour line segment	74, 86, 97
C_*	final contour image resulting from contour extraction	75, 85
$\mathrm{C}_\sqcup$, $\mathrm{C}_\sqcap$	contour image resulting from the use of edge threshold $\tau_\sqcup$ or $\tau_\sqcap$	85
c^+, c^-	left or right end points of a contour line segment c	90, 97
$\lvert c\rvert$	length (i.e. number of pixels) of a contour line segment	75, 86, 97
D	difference image	73, 84
$\mathbf{d}$	differential disparity between background and foreground	75, 97
$\mathbf{d}_\mathrm{B}$	background disparity	67
$\mathbf{d}_\mathrm{F}$	foreground disparity	70, 98
$\bar{\mathbf{d}}_\mathrm{F}$	average foreground disparity	98, 102
$[\mathbf{d}^i_{min}, \mathbf{d}^i_{max}]$	i^{th} disparity range of interest in the disparity histogram	75
$\tilde{\mathbf{d}}$	weighted average disparity	86
$\boldsymbol{\delta}^\top$, $\boldsymbol{\delta}^\bot$	distance from top or bottom end of a contour region to another region	89
E	edge image	73, 84
e^+, e^-	positive or negative edge points	74, 84
ε_E	minimum detectable intensity of edge points	74, 85
$x = \mathbf{f}^+(y)$	line joining the left extrema of the end segments of a contour region	90
$x = \mathbf{f}^-(y)$	line joining the right extrema of the end segments of a contour region	90

$\bar{\mathbf{i}}$	average intensity	86
$\hat{\mathbf{i}}$	summed intensity	88
κ, ς	tunable parameters for computing ${\tau_{\mathrm{X}}}^i$	77
$\boldsymbol{\mu}_{\mathrm{C}}$	convexity measure	90
$\boldsymbol{\mu}_{\mathrm{D}}$	disparity contour direction of a contour region	101
$\boldsymbol{\mu}_{\mathrm{I}}$	integrated intensity/disparity measure	87, 89
$\boldsymbol{\mu}_{\ltimes}$, $\boldsymbol{\mu}_{\rtimes}$	foreground confidence of the left or right neighborhood	101
$\boldsymbol{\mu}_{\mathrm{M}}$	matching cost between two contour regions or of an object	102, 103
$\boldsymbol{\mu}_{\natural}$	overlap measure between two contour regions	102
$\lvert\mathrm{O}\rvert$	number of contour regions in an object	98
p_j^i	j^{th} peak in the i^{th} horizontal signature	75
p, q	image pixel or point	85, 86
$\boldsymbol{\pi}_y$	projection on the y axis	102
R	contour region formed by connected contour line segments	85, 86, 98
R^*	contour region with the largest $\hat{\mathbf{i}}$	88
$\mathrm{R}_{\sqcup}$, $\mathrm{R}_{\sqcap}$	contour region resulting from the use of edge threshold $\tau_{\sqcup}$ or $\tau_{\sqcap}$	85
$\lvert\mathrm{R}\rvert$	number of contour line segments in a contour region	86, 98
$\mathrm{R}_{\bowtie}$	grouping candidate of R	103
$\mathrm{R}_i^{\top}$, $\mathrm{R}_i^{\perp}$	contour region to group with R_i at its top or bottom end	89
S	set of contour regions in a contour image	87
s	background scene point	66
S_*	set of contour regions resulting from contour outlier removal	103
$\mathrm{S}_{*\mathrm{D}}$	set of contour regions after disparity verification grouping	103
$\mathrm{S}_{*\mathrm{I}}$	set of contour regions after integrated intensity/disparity grouping	103
$\mathrm{S}_{\sqcup}$, $\mathrm{S}_{\sqcap}$	set of contour regions resulting from the use of edge threshold $\tau_{\sqcup}$ or $\tau_{\sqcap}$	85
$\lvert\mathrm{S}\rvert$	number of contour regions in a region set	87
$\mathrm{S}_i^{\top}$, $\mathrm{S}_i^{\perp}$	set of candidate regions for grouping with R_i at its top or bottom end	89

σ_{d}	deviation of disparity	86
$\hat{\sigma}$	deviation of summed intensity	88
σ_{i}	deviation of intensity	86
$\tilde{\sigma}$	coupled deviation of intensity and disparity	87
τ_{E}	threshold on relative intensity of edge points	74, 85
$\tau_{\bowtie}$	distance threshold for defining the neighborhood of a contour region	100
$\tau_{\sqcup}$, $\tau_{\sqcap}$	low or high threshold on relative intensity of edge points	85
τ_{M}	matching cost threshold	103
$\tau_{\natural}$	overlap threshold	102
$\tau_{\mathrm{X}}{}^{i}$	distance threshold for the i^{th} horizontal signature	76
$\mathrm{V_L}$, $\mathrm{V_R}$	original left or right view after preprocessing	67
$\mathrm{VDM_B}$	view difference map by background hypothesis falsification	67
$\mathrm{VDM_F}$	view difference map by foreground hypothesis verification	99
$\mathbf{x_L}$, $\mathbf{x_R}$	x coordinate in left or right view	66, 67
$\mathbf{y}$	y coordinate	66

CHAPTER 1
Introduction

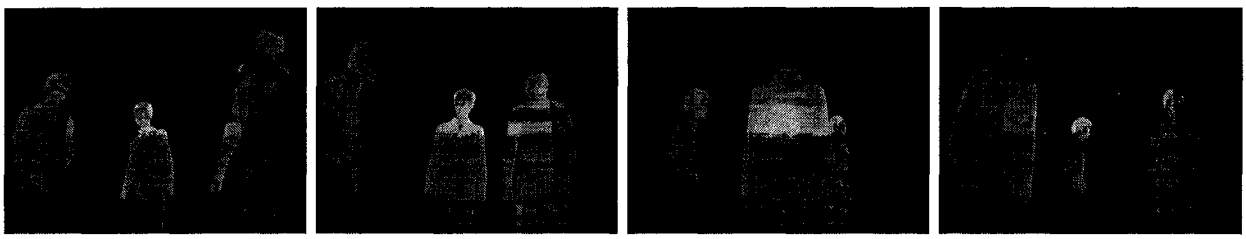
Extracting moving objects from a static or dynamic scene is useful for a wide variety of applications, such as video surveillance, tracking, or event recognition. The problem of foreground/background separation in a dynamically textured scene is particularly interesting to virtual reality environment research, and the entertainment and film industry, where projected video backgrounds are often present. Efficient, high-quality foreground object location and segmentation is also an important research topic in itself and has been studied for decades in the computer vision community. It is a useful preprocessing step for more general image content analysis in attentional contexts, allowing computational effort to be focused on foreground objects.

Most segmentation systems use information from a single camera, and may be classified into two general categories: background suppression and object-specific foreground extraction.

Background suppression is commonly used for detecting moving objects, where each image is compared against a reference or background model [67, 130, 146, 42, 27, 175]. Simple background suppression, such as background subtraction and frame differencing, performs poorly due to illumination change and shadow; see examples shown in Fig. 1–1, 1–2, 1–3, and 1–4. Therefore, statistical models are often used to represent background pixel values, to adapt to appearance variations. The most popular background models include per-pixel single Gaussian model [165], Gaussian Mixture Model [145, 146], linear prediction [155], eigenbackground model [118], Hidden Markov Model [130], optical flow [64], and median filtering [42]. Most of the background models proposed focus on the texture of the background. Despite their adaptability to slow changes in lighting, texture, geometry, shadow, and repetitive background motion such as tree leaves and ripples, they all assume that such changes in the background are significantly less dynamic than

(a) Image of a static background.

(b) Samples from an image sequence with static background.

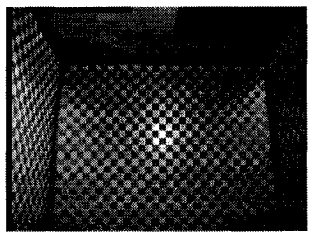


(c) Results of background subtraction.

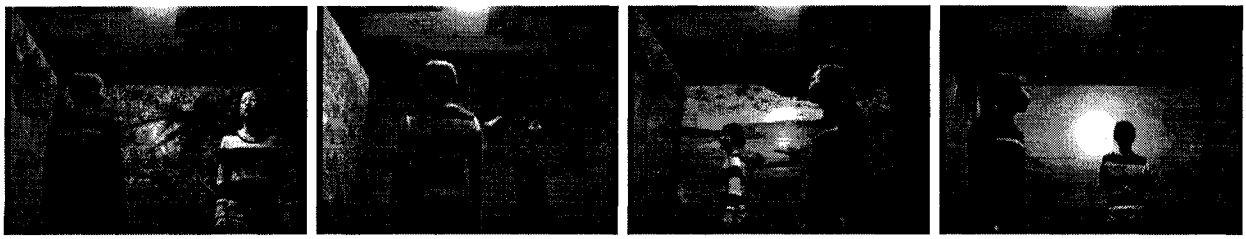
Figure 1–1: Background subtraction on an image sequence with static background.

those of the foreground. Problems related to truly dynamic environments, such as virtual reality environments or modern commercial displays, have not been addressed.

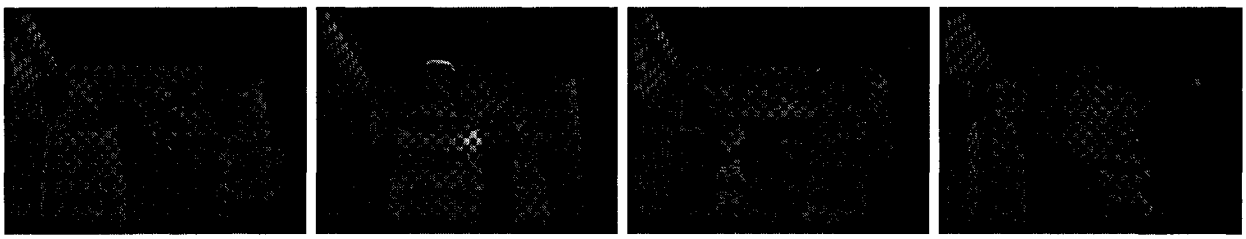
An example of such dynamic environments is the *Shared Reality Environment* (SRE) and its conceptual prototype is illustrated in Fig. 1–5. The aim of the SRE project is to create an audio and video enhanced virtual environment with a purpose of facilitating interactions between users regardless of distance. In order to successfully immerse real objects, such as humans, in a virtual world, it is necessary to separate them from their actual physical surroundings, which may contain fast background motions as shown in Fig. 1–2(b). Therefore, the assumption of static or slowly changing background is no longer valid here and a real-time object segmentation system that is able to tackle dynamically textured scenes is required.



(a) Background image.



(b) Samples from an image sequence with projected video background.



(c) Results of background subtraction.

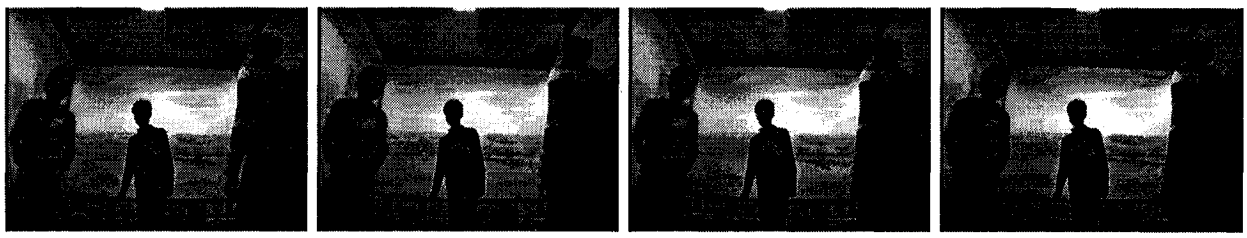
Figure 1–2: Background subtraction on an image sequence with projected video background.

A foreground object can be detected directly from a scene based on an object-specific foreground model, without relying on the knowledge of the background. The information used for foreground modeling can be collected from texture [14, 13, 155, 80, 140], shape [24, 114, 48, 168, 174], motion [15, 142, 119, 51], or a combination of the three [165, 16, 67, 150, 87]. Texture modeling includes the per-pixel Gaussian model [165], histograms [155], Principal Component Analysis [14, 13] and wavelets [80, 140]. Shape modeling is often based on contours [16]. Motion is usually expressed by dynamic models [165, 16, 67]. Object-specific modeling limits a segmentation system to specific targets. A new modeling process is required whenever a new object is involved, which may be time-consuming and impractical for online processing.

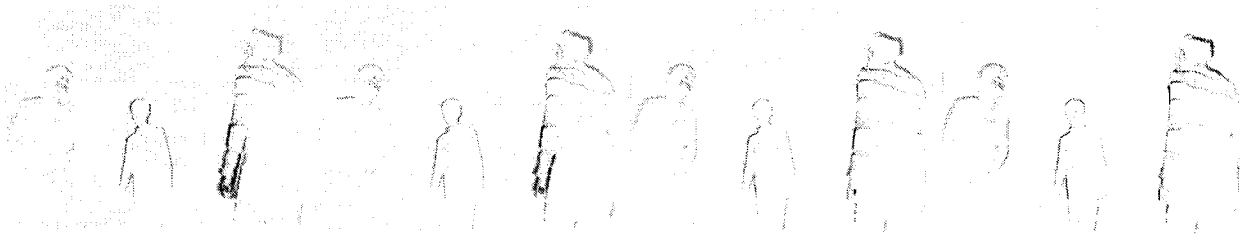
Layered motion segmentation has become popular in the past decade. It decomposes an image sequence into a set of overlapping layers with each layer's motion described by a smooth

(a) Sample image sequence with static background.

(b) Results of frame differencing.



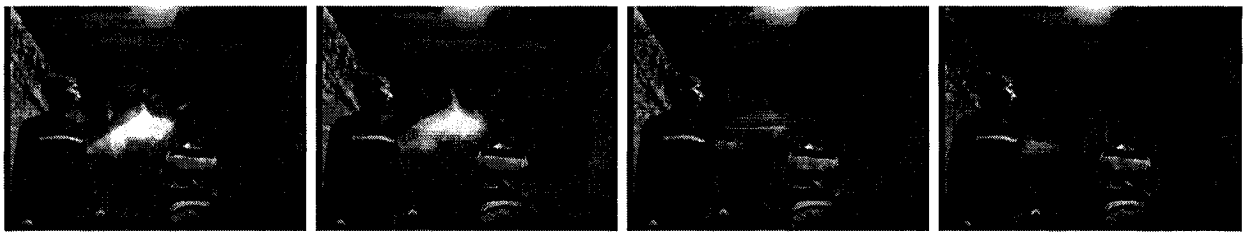
(a) Sample image sequence with static background. (cont.)



(b) Results of frame differencing. (cont.)

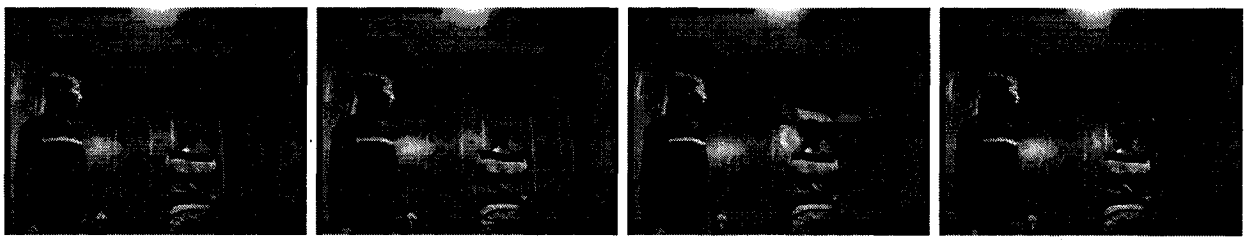
Figure 1–3: Frame differencing on an image sequence with static background, captured at 30 frames/sec. (White denotes zero difference.)

optical flow field. The discontinuities in the description are attributed to object occlusions, resulting in a 2.5D representation of the scene that consists of motion layers as well as occlusion and depth ordering. The most representative work on layered motion segmentation includes that of Wang and Adelson [160], Jepson and Fleet [78], Hsu et al. [73], Ayer and Sawhney [3], Weiss and Adelson [162], and Xiao and Shah [166]. Unfortunately, these methods almost all involve the



(a) Sample image sequence with projected video background.

(b) Results of frame differencing.



(a) Sample image sequence with projected video background. (cont.)

(b) Results of frame differencing. (cont.)

Figure 1–4: Frame differencing on an image sequence with projected video background, captured at 30 frames/sec. (White denotes zero difference.)

computation of optical flow, which is time-consuming and impractical for real-time use. They are also unable to distinguish between a real scene and the projected video of a scene.

Using information from two or more cameras is a biologically motivated and attractive alternative for tackling dynamic environments, which may also offer benefits in dealing with occlusion. Some researchers use multiple cameras mainly to cover a large field of view with little overlap between different views [29, 2]. Such setups do not make use of stereo information, and

Figure 1–5: A conceptual prototype of a node of the Shared Reality Environment.

their most important problems to solve are matching and switching between different camera views.

In order to obtain more accurate 3D information, multiple cameras with overlapping fields of view have been used. Wide baseline setups [32, 47, 89, 107] have difficulties in stereo matching and uncertainty in object correspondence across multiple views. By exploring the potential of multiple calibrated cameras with a small baseline, some systems work from an extensive 3D reconstruction to object segmentation and tracking [77, 112], for which the good performance of a stereo algorithm is crucial. However, frame-by-frame stereo reconstruction is expensive and so far unsuitable for real-time or interactive applications. Existing stereo matching algorithms [116, 131, 93, 149] also tend to be very sensitive to differences in camera response; obtaining sufficiently well-matched cameras may pose a practical difficulty. Finally, the uniform or repetitive textures common in indoor scenes and video-augmented spaces constitute worst-case inputs for stereopsis, often leading to disappointing results.

Some systems combine stereo matching and segmentation [5, 154, 11, 99]. Despite their good performance, the high computational cost remains a weakness of these methods. Recently, both *layered dynamic programming* and *layered graph cut* methods [91] have been proposed for bi-layer foreground-background segmentation. Both methods yield comparably high accuracy in their experimental results and have the potential for real-time applications. However, they have been tested only on images with a large depth difference between background and foreground.

Although dynamic backgrounds such as people walking across the room were tested, foreground motion was restricted to a small depth range, close to the camera, which does not suit most real-world applications. Although a few real-time stereo systems have been claimed in the past decade [85, 110, 55], these systems rely on the computational power of multiple processors and give various definitions of "real-time". None of them addresses the issue of dynamic background.

This thesis presents a stereo-based object segmentation system that incorporates the simplicity and efficiency of the background subtraction approach and the capacity to handle dynamic lighting and background texture, should they occur, as well as static background environments. The method proposed here does not rely on full stereo reconstruction, but introduces *disparity contours* to separate multiple objects at different depths, and is applicable to large textureless regions. The system demonstrates near real-time performance and has the potential of providing precise object contours rather than just bounding boxes. It is also extensible to perform 2D and 3D object tracking.

The proposed stereo-based segmentation system uses a pair of calibrated cameras with a small baseline and factors the segmentation problem into two stages: a well-understood offline stage and a novel online one, as illustrated in Fig. 1–6.

The offline stage first performs a camera calibration to estimate camera parameters. Based on the calibrated parameters, a disparity map of the background scene is constructed to store pixel correspondence information between the left and right views of the background. Since the *background disparity map* (BDM) is determined by the 3D geometry of the background, which is likely to be more stable over time than texture and illumination, the BDM representation is valid over long term even in the presence of lighting and texture change.

The online processing stage compares corresponding new frames synchronously captured by the two cameras according to the BDM in order to falsify a background hypothesis of the scene, and stores the mismatch information in a *view difference map* (VDM), which enables the rapid extraction of object boundary contours that encode depth information. Object segmentation is

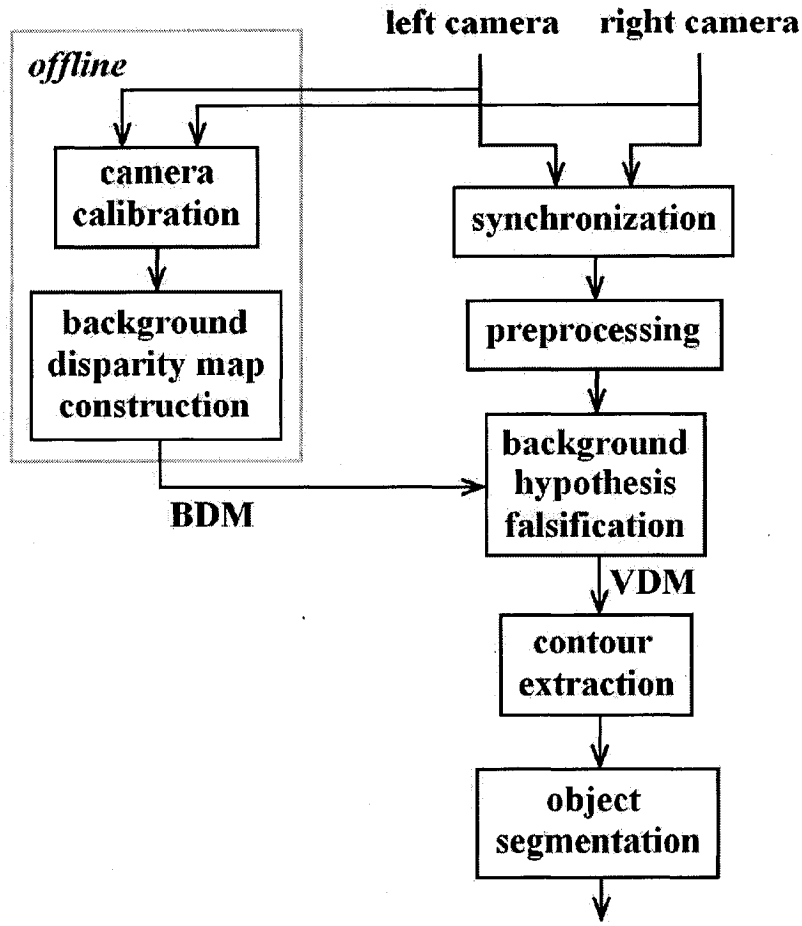


Figure 1–6: Stereo-based object segmentation system overview.

conducted based on the extracted contours. Of course at the beginning of each online processing loop, a preprocessing step is necessary for image distortion removal and rectification.

This thesis is structured as follows. Chapter 2 reviews existing approaches to various components of the proposed system, including camera calibration, computational stereo, and object segmentation. Chapters 3 to 7 then provide the detailed explanation of each component of the system and the experimental results, as outlined below. Chapter 3 presents some useful empirical observations from the work on camera calibration. Appendix A briefly describes synchronized video acquisition and image preprocessing. Chapter 4 explains the idea of *background hypothesis falsification* and introduces the concept of *disparity contours*. Chapters 5 and 6 present two initial approaches to disparity contour extraction and object segmentation, with qualitative and

quantitative analysis of experimental results. Chapter 7 explores in more depth the features of *disparity contours* and proposes a much improved method to overcome the weaknesses of the previous two. Finally, Chapter 8 summarizes the work and suggests future directions for this research.

CHAPTER 2
Literature Review

Stereo-based object segmentation involves several areas of research: camera calibration, computational stereo, and object segmentation. This chapter reviews each of them in detail.

2.1 Camera calibration

Camera calibration has received increased attention in the computer vision community during the past two decades [59, 132]. The decreasing cost of computers and cameras has also brought rapid growth in stereo-based applications. In consequence, an ever-increasing population of researchers is coming to depend on camera calibration for their projects.

The purpose of camera calibration is to estimate the parameters of a camera model so as to establish the correspondence between 3D scenes and 2D images captured by the camera, as shown in Fig. 2–1(a). Given a minimum of two calibrated cameras, the 3D structure of a scene can be reconstructed from the 2D information in the camera views. Therefore, camera calibration is a critical first step for stereo applications such as view synthesis, 3D object segmentation and tracking.

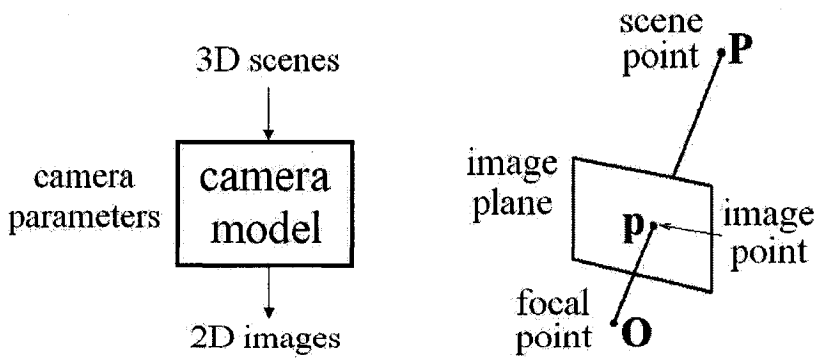


Figure 2–1: (a) Diagram of a camera model. (b) A pinhole camera model.

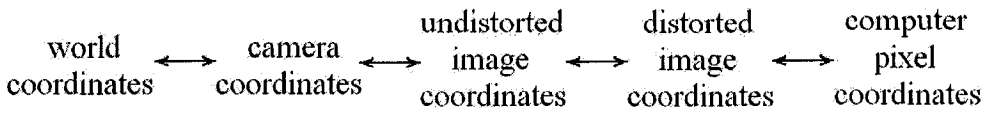


Figure 2–2: Coordinate transformations in a camera model.

2.1.1 Camera models

A camera is usually represented by a pinhole model that maps 3D scenes to 2D images, as illustrated in Fig. 2–1(b). This process conceptually consists of four coordinate transformations, as shown in Fig. 2–2. The first transformation between 3D world coordinates and 3D camera coordinates is determined by camera position and orientation, known as the *extrinsic parameters.* The next three transformations from camera coordinates to pixel coordinates are controlled by internal camera configuration such as the focal length and the image center location, known as the *intrinsic parameters.*

The choice of camera model in various calibration methods varies mainly in characterization of lens distortion [143]. Types of lens distortions commonly considered are radial and tangential. Two common radial distortions are pincushion and barrel distortions. Tangential distortions are usually caused by decentering or thin-prism distortion. One of the effects of tangential distortion is that a straight line passing through the center of the field of view may appear in the image as weakly curved line.

Among existing calibration methods, Tsai used a second order radial distortion model [158] while Zhang employed both the second and fourth order terms [172]. Heikkilä included two further decentering distortion components [71], while Lavest et al. added the sixth order radial term [95]. Weng et al. incorporated the thin prism distortion whose coefficients could be merged with those of the decentering distortion in actual calibration [163]. Most camera models assume zero skewness, i.e., the angle between x and y image axes is 90° [158, 71, 163], but Lavest [95] and Zhang [172] estimate skewness as a variable.

Although the significance of each distortion coefficient apparently depends on the actual camera and its distortion characteristics, it is not clear which distortion coefficients most contribute to the accurate calibration of various cameras with different distortion characteristics, whether the addition of higher-order radial distortion components or decentering distortions actually improves calibration accuracy and if so, to what degree.

2.1.2 Calibration algorithms

According to the nature of the training data used, existing calibration methods can be classified as coplanar or non-coplanar. The coplanar approaches perform calibration on data points limited to a single planar surface. These methods are either computationally complex or fail to provide solutions for certain camera parameters, such as the image center, the scale factor, or lens distortion coefficients [33]. In contrast, the non-coplanar approaches uses training points scattered in 3D space to cover multiple depths, and do not exhibit such problems.

Non-coplanar approaches fall into a number of categories. World-reference based calibration is a conventional approach requiring the 3D world coordinates and corresponding 2D pixel coordinates of a set of feature points [54, 158, 163, 71, 132]. The disadvantage of this approach is that either a well-engineered calibration object is required, or the environment has to be set up carefully to achieve accurate 3D measurements. Geometric invariant based methods use parallel lines and vanishing points as calibration features. Although world coordinate measurement is not required, special equipment may be necessary for measuring certain variables, for example, the ratio of focal lengths on the two axes [31]. Implicit calibration methods [161] have no explicit camera model and may achieve high accuracy, but they do not reveal the physical parameters of a camera and are computationally expensive because of the large number of unknown variables involved. Auto- or self- calibration approaches determine camera parameters directly from multiple uncalibrated views of a scene by matching corresponding features between views, despite unknown camera motion and even changes in some of the intrinsic parameters [53, 68]. Unfortunately, due to the difficulty of initialization [59], auto-calibration results tend to be unstable [17]. Planar auto-calibration [156] addresses this initialization difficulty by using multiple views of a

planar scene, taking advantage of the fact that planes are simple to process and allow reliable and precise feature or intensity-based matching. Sturm-Maybank [147] and Zhang [172] both extended this idea by taking into account the relative geometric information between planar feature points, with Zhang's method estimating lens distortion coefficients, a factor not included in the former work. According to Sturm and Maybank's singularity analysis [147], degenerate situations can easily be avoided. Another extension of the multi-view planar approach suggests the use of angles and length ratios on a plane but provides no experimental results [98].

It is worth noting that these multi-view planar calibration methods differ from the coplanar methods reviewed by Chatterjee [33]. The methods described here rely on a planar calibration pattern positioned at various orientations. They exploit the co-planarity constraint on the points in each sample set to reduce or eliminate the need for 3D measurement, but still sample a 3D region. Similarly, a one-dimensional object can be positioned at various spots with various orientations in a 3D space to provide non-coplanar points for calibration [173].

While it is trivially obvious that more accurate training data will result in better calibration, the more salient issue is what accuracy can be achieved within the practical limits of most research environments. The impact of noise on calibration accuracy has been studied by Lavest et al. [95] and Zhang [171], however, their experiments on real data involved calibration points with small distance coverage in 3D space and did not use a separate test data set to verify the scalability of calibrated results. This apparent gap in the literature motivates the work reported in Chapter 3.

2.2 Computational stereo

Computational stereo for extracting three-dimensional scene structure has been studied for decades [46, 83, 137, 26]. Its basic principle is stereo triangulation, locating a single 3D point from a unique pair of image points in two observing cameras [157], as illustrated in Fig. 2–3. The primary problem to solve is stereo correspondence, i.e., computing the displacement (*disparity*) of a projected point in one camera view with respect to the other. Stereo matching algorithms, aiming to solve the correspondence problem, must generally address three issues: token types, source of constraints, and support or compatibility aggregation.

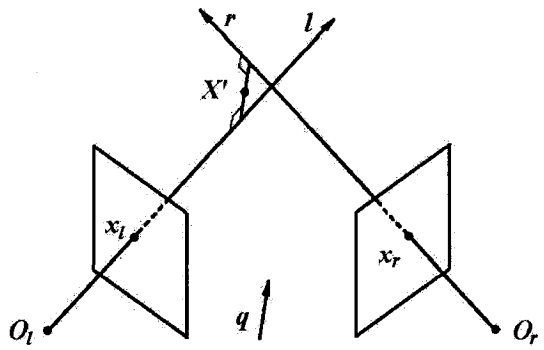


Figure 2–3: Stereo triangulation.

2.2.1 Tokens

Tokens are the information extracted from images, based on which stereo matching is performed. The simplest tokens used in stereo matching algorithms are brightness in gray-scale images or colors in color images [62, 25, 109, 148]. The disadvantage of using intensity values directly is that they are sensitive to illumination, contrast, and differences in viewing positions of two cameras.

Feature-based stereo algorithms use symbolic tokens derived from intensity images, which are more stable to changes in contrast and ambient lighting. The most commonly used low level features are edges or points of interest [106, 63]. The main disadvantage in using low level features for matching is that the hundreds of edge points demand a large number of computations and result in a large number of ambiguous candidate matches.

The most common high level tokens include line segments [121], contours [167], curves [138] and regions [61, 133, 126]. The use of complex image features has two main advantages in matching. First, they are usually fewer in number, resulting in much faster correspondence algorithms. Second, they contain rich descriptive attributes, such as length, width, area, global intensity information and contrast. This attribute information can reduce the number of match candidates and significantly increase the matching accuracy. These advantages of high level tokens are desirable for real-time stereo systems.

2.2.2 Constraints

Matching constraints have been classified by Jones [83] into unary and binary. Unary constraints contain information specific to each match, while binary constraints measure the compatibility of pairs of matches. This classification is essentially based on the level of locality or globality of a constraint. Here we refer to individual constraints and compatibility constraints.

Individual constraints. One of the most important classes of individual constraint is that of geometry, the best known of which is epipolarity. The epipolar constraint reduces the search for a corresponding point from an entire 2D image to a 1D epipolar line [157]. This reduction in search space significantly improves matching speed and reliability, and has been adopted by many stereo algorithms. To take advantage of the epipolar constraint, image rectification techniques have been developed to virtually modify the camera parameters so that pairs of conjugate epipolar lines become co-linear and parallel to the horizontal image axis [60]. Often used in conjunction with the epipolar constraint is the view volume constraint, which limits the length of the epipolar line to the width of the image plane.

The similarity constraint is another important individual constraint, and forms the basis of stereo matching. Ideally, best matches can be found simply by locating corresponding tokens that satisfy the similarity constraint. The similarity between two pixels or their neighborhoods is measured through a matching cost computation. Disparities are obtained either by the winner-take-all method directly from the matching costs or by an optimization scheme based on a global cost.

The most common pixel-based matching costs include the *sum of squared differences* (SSD), the *sum of absolute differences* (SAD), which is often used for computational efficiency, and the *normalized cross correlation* (NCC) [137, 26].

$$SSD: \quad \sum_{u,v} \left(I_1(u,v) - I_2(u+d,v)\right)^2 \tag{2.1}$$

$$SAD: \quad \sum_{u,v} \left|I_1(u,v) - I_2(u+d,v)\right| \tag{2.2}$$

$$NCC: \quad \frac{\sum_{u,v}\left[\left(I_1(u,v)-\bar{I}_1\right)\cdot\left(I_2(u+d,v)-\bar{I}_2\right)\right]}{\sqrt{\sum_{u,v}\left[\left(I_1(u,v)-\bar{I}_1\right)^2\cdot\left(I_2(u+d,v)-\bar{I}_2\right)^2\right]}} \tag{2.3}$$

Other cost computation methods, e.g., gradient-based measures [135] and non-parametric measures such as rank and census transforms [169], are insensitive to difference in camera gain or bias. The rank transform of a pixel in its neighborhood region is defined as the number of pixels in that region whose intensity is less than that of the center pixel. The resulting values are based on the relative ordering of pixel intensities, which preserves the sharpness of region boundaries. However, because of the information loss of the rank transform, the discriminatory power of the succeeding matching procedure is reduced. The census transform preserves the spatial distribution of ranks by encoding them in a bit string. Matching is then performed using Hamming distance, i.e., the number of bits that differ, between bit strings. It is also possible to correct for different camera characteristics by performing a preprocessing step for bias-gain or histogram equalization [62, 41]. Additional proposals include phase and filter-bank responses [106, 81, 103, 100].

Unfortunately, all these more robust methods require extra processing compared to the simple pixel-based matching cost computation and negatively impact performance.

Compatibility constraints. Compatibility constraints express the support for the match of a token from the matches of the token's neighborhood. Such constraints include continuity/smoothness, coherence, uniqueness and topology.

The continuity constraint rests on the observation that points adjacent in 3D space remain adjacent in each image projection. Therefore, disparity varies smoothly on object surfaces, along edges of surfaces, and along contours. To realize this, adjacent disparities must satisfy a disparity gradient limit [28, 120], lie on the same edge contour or line segments [106, 63], or satisfy a smoothness term in a global energy function [141, 136, 104, 131, 152, 23, 22, 92, 11, 99]. This is a very useful constraint for environments with opaque objects.

The coherence constraint recognizes the case of transparent objects [124]. It allows the occurrence of a discontinuous disparity field if it is the result of several interlaced continuous disparity fields, each corresponding to a piecewise smooth surface.

The uniqueness constraint requires a one-to-one mapping, i.e., each token in an image can be assigned to one and only one disparity value [92]. In the case of ambiguous matching, the uniqueness constraint is usually implemented by further examining the multiple tokens belonging to a single match.

The topological constraints are based on the fact that the projections of a 3D structure in different viewpoints have the same topology. Popular among these are the ordering constraint, parallelism, and the relative position constraint. The ordering constraint preserves the order in which 3D events are projected [116, 40, 74, 141]. Parallelism means that nearby parallel lines in 3D space must be projected to nearly parallel lines in each 2D view [82]. The relative position constraint assumes that the relative positions of tokens remain similar between images [72]. This is less true when features are widely separated in a 3D space where occlusions exist.

The compatibility constraints can be combined through a weighted average of individual sources [82, 141, 11, 23] or a series of thresholded filters [72]. Most compatibility constraints are implemented in a heuristic manner within various optimization methods to achieve a global compatibility [83].

2.2.3 Compatibility or support aggregation

The distribution of the constraint information discussed above is usually sparse in an image, which leads to poor or ambiguous matching results. Therefore, compatibility needs to be propagated from reliable to unreliable matching regions to improve results. Almost all stereo algorithms consider compatibility between neighboring regions. Their difference lies in the degree of globality of the compatibility considered. Accordingly, stereo algorithms can be divided roughly into local and global approaches.

Local compatibility aggregation. Local correspondence methods, such as block matching, gradient methods, and feature matching, consider small neighborhoods.

Intensity based block matching methods estimate disparity at a point in one image by comparing a small region (the template) around that point with a series of small regions in the other image (the search region). As stated in Section 2.2.2, the epipolar constraint greatly

reduces the search space to one dimension. Matching cost computation, reviewed in that section, is the key to the block matching approach. The biggest challenge of using block matching is how to balance speed and accuracy; both are affected by the size of the template. A small template can yield fast yet ambiguous matching, whereas increasing the template size results in slower matching that is more robust to noise. The naive implementation of any block matching method is inefficient due to redundant computations. Somewhat faster algorithms have been developed by Sun [148], Muhlmann et al. [109] and Schmidt et al. [139]. However, despite its simplicity, point-to-point matching to determine view correspondence is inherently slow and unreliable. More constraints are necessary to restrict the search space.

Gradient-based or optical flow methods [102] determine small local disparities between two images by formulating differential equations relating motion and image intensity. Assuming that the intensity of a point in the scene is constant between two views, the horizontal translation of a point from one image to the other is computed by a differential equation. Assuming that disparity varies smoothly over a small window of pixels, the disparity at a point may be estimated using least squares on a system of linear differential equations at each pixel in an n pixel window around that point. Gradient-based methods are suitable for estimating small disparities. However, the disparity range of a stereo image pair is typically much larger than one pixel, limiting the method's performance.

Intensity or gradient-based methods are sensitive to depth discontinuity. Feature-based methods seek to overcome this problem by limiting the regions of support to specific reliable features in the images, such as edges, curves, etc. Three classes of feature-based approaches are worthy of mention: hierarchical geometric feature matching, segmentation matching, and wavelet analysis.

Hierarchical geometric feature matching algorithms [105, 159] often take a top-down approach and exploit several types of features, from low-level lines, vertices and edges to high-level regions or surfaces. Matching begins at the highest level of the hierarchy (regions or surfaces) and proceeds to the lowest (lines). This allows coarse, reliable features to provide support for

matching finer, less reliable features and reduces the computational complexity of matching by reducing the search space at lower levels. Segmentation matching takes either a bottom-up approach as is done by Sander et al. [133], who grows regions from pixels, or a top-down approach as is done by Randriamasy and Gagalowicz [126], who recursively divide regions using a thresholding method based on contrast minimization. These approaches are usually sensitive to the quality of the segmentation. Wavelet analysis methods [103, 100] represent images using a multi-scale decomposition. A hierarchical disparity propagation scheme is used to aggregate matching results at each level.

The disadvantage of feature matching is its difficulty in generating dense depth maps. However, its benefits include increased speed and accuracy, especially when used with high-level tokens, as mentioned in Section 2.2.1.

Global compatibility aggregation. Global compatibility aggregation aim to reduce the sensitivity of a stereo algorithm to occlusion and uniform texture. Some of the most popular approaches include graph isomorphism, energy optimization, 3D geometry approximation, and structured light.

Graph isomorphism The tokens in an image can be represented by a graph structure whose nodes represent individual tokens and whose links encode topological or spatial relations between tokens. Such graphs are referred to as relational or attributed graphs [45]. The reversible mapping between all nodes of one graph and all nodes of another is known as *isomorphism.* In stereo correspondence problems, image graphs are likely to be incomplete due to occlusion, hence a *double subgraph isomorphism* is performed to locate a mapping between the two subgraphs [20].

The most commonly used algorithms are based on recursive graph search [6], which explores all or a subset of potential mappings between two graphs to minimize some distance measures [45, 20]. These methods are often feature-based and construct interpretation trees whose root-to-branch paths represent all possible interpretations of the correspondence problem [61]. In order to restrict the interpretation tree to a computationally feasible size, some methods build the tree based on a set of reliable seed matches of some minimum size [61].

Energy optimization In global energy optimization approaches, a cost or energy function is defined by combining a variety of constraints. Optimization is performed to maximize or minimize this energy function. Here we review some of more recent approaches including dynamic programming, graph cut, nonlinear diffusion, and belief propagation.

Based on the epipolar monotonic ordering constraint, dynamic programming [39] optimizes a global cost function by finding the minimum cost path through a *disparity-space image* (DSI) [74]. The total cost is summed up recursively from the costs of partial paths, and the cost of each point in the DSI is defined using the similarity measures in Section 2.2.2. There are two ways to construct a DSI. The axes of a DSI can be defined as the left and right scanlines [116, 40], as shown in Fig. 2–4(a), or the left scanline and the disparity range [75], as in Fig. 2–4(b). In order to reduce ambiguity, various methods have integrated interscanline constraints to minimize the sum of costs over two-dimensional regions [4, 116, 8, 40, 10]. The major disadvantage of dynamic programming is the possibility that local errors may be propagated along a scanline, corrupting other potentially good matches [26].

A variation of dynamic programming uses intrinsic curves instead of the DSI [153]. An intrinsic curve is a vector representation of image descriptors defined by applying operators, e.g, edge and/or corner operators, to the pixels in a scanline. Thus, the disparity search problem

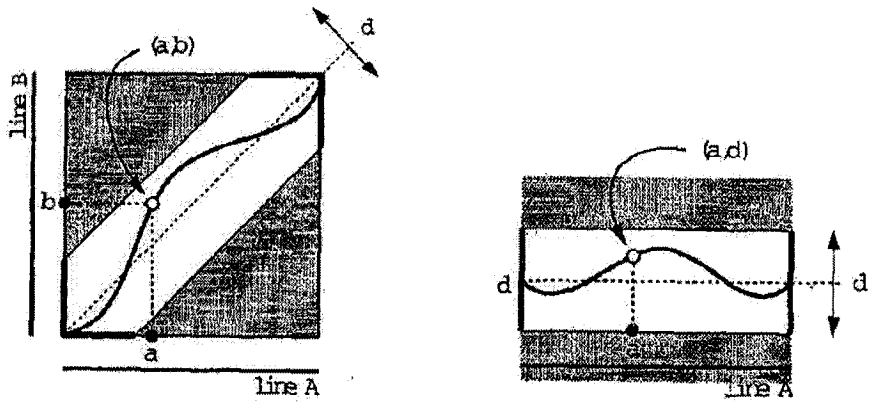


Figure 2–4: Disparity-space image with (a) axes defined as the left and right scanlines, and (b) axes defined as the left scanline and the disparity range. Fig. 2 of Roy and Cox 1998 [131].

becomes a nearest neighbor problem in intrinsic curve space. Although this is an interesting idea, no advantage over the DSI approach is demonstrated in experimental results.

The most significant limitation of dynamic programming is its inability to incorporate strongly both horizontal and vertical continuity constraints. Graph cut approaches first proposed by Roy and Cox [131] extend DSI construction from a 2D left-disparity representation to a 3D left-disparity representation as in Fig. 2–5(a). Based on the 3D DSI, a directed graph as shown in Fig. 2–5(b) is defined, where the vertices represent all possible matches, and each edge has an associated flow capacity defined as a function of the costs of the adjacent nodes it connects. The stereo problem becomes the problem of finding the cut with the minimum capacity that maximizes the flow through the graph. The standard *relabel-to-front* algorithm [39] was used by Roy and Cox [131] to solve the *max flow* problem. An efficient data structure has been introduced by Thomo et al. [152] to reduce the memory space, making large data sets more manageable. Boykov and Kolmogorov [22] developed an approximate Ford-Fulkerson style augmenting paths algorithm, which they showed to be much faster in practice than the standard *relabel-to-front* approach. Boykov et al. [23] proposed a multi-way cut graph structure, as demonstrated in Fig. 2–6, instead of the 2-terminal graph and used α-*expansion* and α-β-*swap* algorithms to modify labels of arbitrarily large pixel sets simultaneously. Kolmogorov and

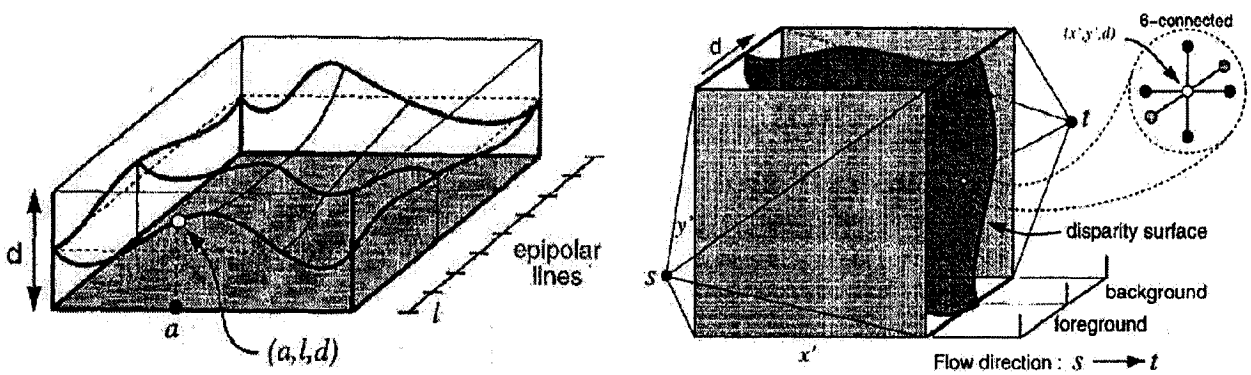


Figure 2–5: (a) 3D disparity-space image with axes defined as the left image and the disparity range between the left and right images. Fig. 3 of Roy and Cox 1998 [131]. (b) A directed graph for solving maximum flow problem. Fig. 4 of Roy and Cox 1998 [131].

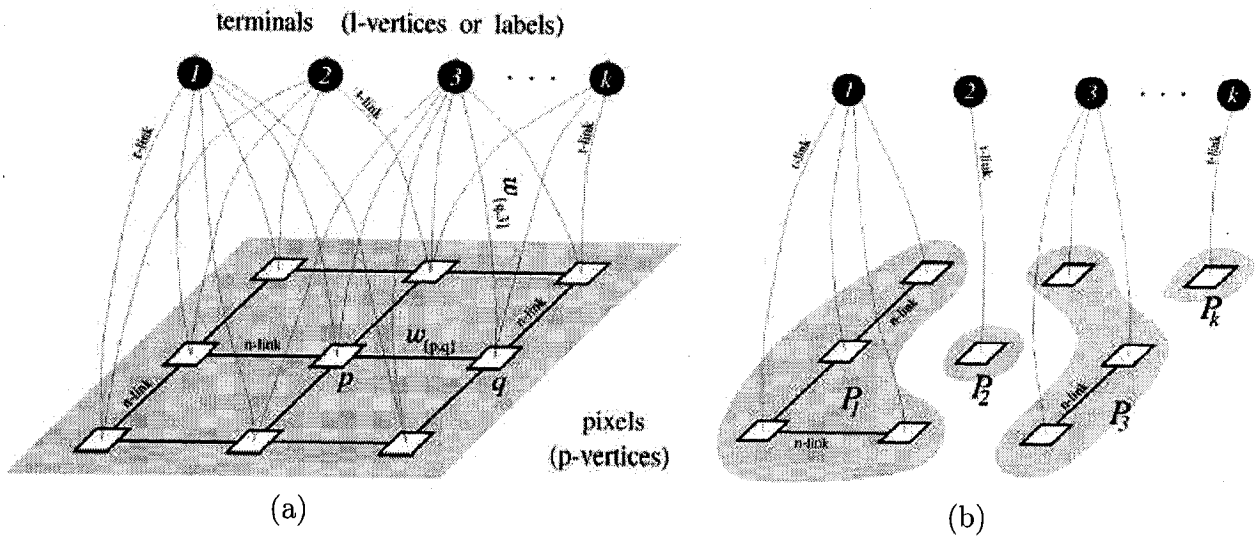


Figure 2–6: (a) Example of a graph with multiple terminals. (b) A multi-way cut on a multi-terminal graph. Fig. 9 of Boykov et al. 2001 [23].

Zabih [92] suggested another graph structure in which the vertices represent pixel correspondence rather than pixels themselves so that a uniqueness constraint can be imposed to handle occlusion.

Other methods considering 2D global optimization include nonlinear diffusion and belief propagation. Instead of using fixed size, rectangular windows to compute neighborhood support as in most block matching schemes, nonlinear diffusion [141, 136, 104] combines various models non-uniformly at locations of ambiguous matches. The belief propagation method [149] uses a Markov Network to represent the posterior probability function over disparity. The standard *max-product* algorithm is employed to maximize the posterior probability (*belief*) and update the network by propagating messages between nodes.

3D geometry approximation Some global methods reconstruct a 3D scene by explicitly modeling its 3D geometry. Fua and Leclerc [58] modeled surfaces as a mesh that is iteratively adjusted to minimize an objective function. Faugeras and Keriven [52] proposed a similar approach but modeled surfaces using level sets. Kutulakos and Seitz [94] represented the scene as a volume and refined the surface using a space carving method by checking voxel consistency between different views. Tomasi et al. used a multi-way cut segmentation approach [11, 99] to

Table 2–1: Performance comparison of best stereo algorithms on 384 × 288 Tsukuba image with 15 disparity labels.

method	processing time (s)	processor
multi-way graph cut [23] (Boykov et al. 2001)	75	500MHz Pentium III
graph cut with occlusion [92] (Kolmogorov and Zabih 2001)	83	500MHz Pentium III
belief propagation [149] (Sun et al. 2003)	51	1.7GHz Pentium IV ($\approx$ 1.2$\sim$1.4GHz Pentium III)
multi-way cut segmentation [99] (Lin and Tomasi 2004)	no data, expected to be slower than others	

minimize an energy function by alternating between an image segmentation using Boykov et al.'s multiway cut algorithm and a surface fitting process using either affine parameters [11] or bicubic B-splines [99].

The recent multi-way graph cut methods [23, 92], the belief propagation method [149], and the multi-way cut segmentation approach [99] are shown to be among the best performers in stereo algorithms according to Scharstein and Szeliski's review [137]. Table 2–1 compares their processing time per image pair on the Tsukuba image of resolution 384 × 288 with 15 disparity labels. Although no data is available, the multi-way cut segmentation method is expected to be slower than the others, as it aims to produce a real-valued subpixel disparity map. As can be seen, none of these methods is close to real-time performance, even on low resolution images.

Structured light Unlike the methods above that rely on stereo triangulation between two camera views, the structured light approach performs optical triangulation between the "views" of a camera and a projector. Instead of matching images captured by two cameras, the image projected (i.e. 'inversely perceived') by the projector and the image perceived by the camera are matched through a space-time analysis [43] to obtain their correspondence.

Various light patterns have been designed to facilitate the matching. Simple ones include a flying spot [128] and a sweeping light plane [84]. Complicated ones include a set of hierarchical gray stripe patterns [134] and time-varying stripe patterns [66]. Color has also been used, such as a color dot pattern [44] or a color stripe pattern [19, 170] with a special color intensity

configuration. Since structured light reduces unambiguous matching on a horizontal scanline, dynamic programming can be used to obtain a global optimum [170].

The benefit of using structured light is its ability to achieve accurate 3D reconstruction of any given shape from the view of a single camera. However, it requires the calibration and synchronization between a camera and a projector, and is only applicable to textureless objects. Although one-shot reconstruction is possible using color stripes [170], multiple images of the same scene are necessary to achieve accurate results, which limits its suitability for moving objects. Although some methods claim to capture moving scenes [19, 44, 66], they impose various constraints on either the appearance or the speed of an object.

2.3 Object segmentation

Object segmentation extracts foreground objects from a background scene, which is a critical early step for many applications of video surveillance and human-computer interaction. Several aspects of object segmentation are reviewed in this section, including sources of information, background suppression, foreground object extraction, tracking, layered motion segmentation, and stereo segmentation.

2.3.1 Sources of information

Information collected for object segmentation can be roughly classified into four categories: texture, shape, motion and depth.

Texture/Appearance. Texture and shape information describe the visual properties of an object. Raw texture information includes the intensity [155, 145, 130, 146, 38] and the color [165, 145, 146, 38, 42, 27, 91] of pixels. Normalized color is often used to reduce its sensitivity to changes in ambient light [165, 27]. Intensity and color can also be combined [144]. Computed texture includes derivatives [130], contrast [91] and reflectance ratio [27].

Appearance modeling is another useful method to describe the texture of an object. The most commonly used appearance model is based on *principal component analysis* (PCA) [14, 13, 118]. A less common appearance model expresses object texture by wavelets [80, 140].

Shape. Shape information can be obtained from three sources: edges, silhouettes and contours. Edges are useful in representing salient features in an image such as corners and curves [114, 48, 168, 87], and can be connected to form object contours for high-level modeling [16, 174].

Some methods use silhouettes to represent object shapes [24, 67]. This can provide global information including size, centroid, major axis, and moments of various orders which can be made invariant to translation, rotation and scale. Object contours, and horizontal and vertical projection histograms can be extracted easily from silhouettes [67].

Contours are most commonly used for shape modeling. For human objects, an ellipsoid contour model is often used to detect head, torso, hands, legs, feet, or even clothing [165, 16, 174]. Contours from silhouettes more usually contain an entire object [150].

Specific to human object segmentation, there are several models to describe the human body, including stick figure models and volumetric models such as generalized cones, elliptical cylinders and spheres [1].

Motion. Motion information expresses how an object moves. The basic motion information is position and velocity, which are usually represented by dynamic models [165, 16, 67].

Optical flow is a powerful tool for motion extraction. Various methods have been proposed to estimate the parameters of an affine image motion model based on optical flow [9, 160, 78, 162]. Using a pre-defined basis set of steerable flow fields, motion discontinuities or boundaries can be detected [56, 12].

PCA is another approach to represent motion [15, 142, 119, 51] and can be applied to optical flow [15, 51]. Different variations of the PCA approach have been employed to model periodical human motions [142, 119].

Depth. Depth cues provide 3D information about a scene and are especially important for dealing with multiple object segmentation and occlusions. Depth information can be extracted from motion by using the temporal coherence of an image sequence [9, 160, 78, 73, 3, 162, 79, 166] or through stereo algorithms [5, 11, 21, 69, 154, 99, 91].

The four types of information above are often combined for object representation. For example, intensity and optical flow are combined for more reliable motion estimation [73] or foreground-background separation [64]. Edge and motion are coupled for boundary detection [114]. A dynamic model can be combined with a contour model and become active contours [16]. More and more frequently, appearance, shape and motion are integrated for segmentation [165, 67, 150, 87]. Recently, depth or stereo information has also become involved [5, 11, 21, 69, 154, 99, 91, 97]. Jepson et al. [79] proposed a 2.5D layered polybone model, which used shape and motion parameters and depth ordering information to include visibility and occlusion for multi-object tracking.

2.3.2 Background suppression

Background suppression is a common approach to identifying moving objects, in which each video frame is compared against a reference or background model. There are two basic approaches in this direction. One is background subtraction [42], where a difference image is computed between the current frame and a reference image representing a background model. Thresholding and morphological filtering are then applied to the difference image to locate objects. The other approach is pixel labeling [67, 130, 146, 27, 175]. Pixels in the current frame that deviate significantly from the background are considered to be moving objects. These 'foreground' pixels are further processed for object localization and tracking. This section reviews the key components of background suppression: background modeling and shadow detection.

Background modeling. Simple background subtraction performs poorly due to illumination change and shadow. To adapt to appearance variations, background is often represented by a statistical model of the intensity or color value of every pixel in an image.

A Gaussian model of YUV color distribution was first proposed by Wren et al. [165] to represent each pixel, whose statistics are updated using a simple information weighting. This method is able to adapt to illumination change in the background. This idea was extended by Stauffer and Grimson, who modeled the recent history of every pixel by a weighted mixture of Gaussian distributions (the *Gaussian mixture model*—GMM), where the weights were updated

by an online K-means approximation [145, 146]. The Gaussian models are sorted according to their probability of representing the background and the few most probable models are chosen for background estimation. The GMM approach has the advantages of representing the multi-modal appearance of the background and adapting to slow changes in lighting, texture, geometry, and repetitive background motion such as that of tree leaves and ripples.

Some systems model background using multi-level processing. Toyama et al. [155] proposed a linear prediction method to estimate the intensity of a background pixel from its history. Prediction coefficients are learned by minimizing prediction error. Multi-scale pixel-level models are used for multi-modal background appearance and model switch occurred at frame-level. Haritaoglu et al. [67] maintained the background by keeping track of the minimum and maximum intensity of a pixel and the maximum intensity difference of the pixel between consecutive frames. The background is updated by two processes: a pixel-based update to adapt to illumination changes and an object-based update to adapt to geometry changes such as added or removed objects. This approach is limited by its inability to handle partial occlusion of a person.

PCA, *hidden Markov model* (HMM), and optical flow have also been used in modeling background. Oliver et al. [118] built an eigenspace model using PCA to describe the range of background appearance due to lighting variation over the day. Eigenbackground subtraction was then performed adaptively to detect moving objects. By exploring temporal continuity, Rittscher et al. [130] modeled the grayscale intensity over time for each pixel location as a single HMM with three types of observations: foreground, background and shadow. The parameters of the model were learned by a Baum-Welch algorithm [125]. Gutchess et al. [64] initialized a background model based on optical flow. A net flow between two frames is defined by subtracting output flow from input flow at each pixel. Pixels with high negative net flow caused by objects leaving are likely to be background.

Other simpler and faster background modeling methods include frame differencing [155] and median filtering [42]. Unfortunately, experiments show that more slowly adapting algorithms tend to have better performance than those that adapt quickly [34].

Shadow detection. Moving shadows can be misclassified as foreground, which may lead to drastic changes in the shape of an object or merging of multiple objects. Some methods reduce sensitivity to shadows by using YUV color and normalizing UV with respect to Y [165]. Some methods use HSV color to separate chromaticity and luminosity explicitly, and a shadow is detected by judging its darkening effect on the background [42]. Other methods combine normalized RGB color and reflectance ratio for shadow detection [27]. A detailed evaluation of several shadow detection methods can be found in the work of Prati et al. [123]

Despite their adaptability to various slow changes in the environment, all the background modeling methods reviewed in this section assume that lighting and texture variations of the background are significantly less dynamic than those of the foreground. Problems related to truly dynamic environments, such as virtual reality environments or modern commercial displays, have not been addressed.

2.3.3 Foreground extraction

Foreground objects can be extracted by background suppression or detected directly from an image based on object-specific foreground modeling.

Object extraction via background suppression. Once a reliable background model is available, foreground object extraction can be performed through either background subtraction or pixel labeling, as reviewed in Section 2.3.2. High-level post-processing is usually necessary to yield a meaningful object interpretation.

Most background suppression algorithms generate foreground blobs after thresholding or pixel labeling. Due to noise and misclassification error, these blobs may be disconnected. Therefore, morphological operations are needed to produce a single region for each object that is guaranteed to be connected [165, 145, 67]. Object detection can then be triggered based on the size, saliency, and motion of the regions [42]. Some methods operate directly on the difference image resulting from background subtraction [36, 7], where object regions can be located using a clustering approach such as mean shift [37, 38].

Some background suppression schemes such as frame differencing yield the contour fragments of a moving object. To compute closed bounding contours from contour fragments relies on the technique of contour grouping, which has been studied for many decades in the area of perceptual organization [49, 50]. Contour grouping constraints have also been employed to complement region-based segmentation [129, 96]. However, reliable contour grouping requires a large amount of computation and is unsuitable for real-time applications. The general problem of extracting the bounding contour of an object of arbitrary shape in a complex scene is essentially unsolved.

Object-specific foreground modeling. A foreground object can be modeled by its texture, shape and/or motion, which enables the object to be detected directly from a scene without relying on knowledge of the background.

Texture modeling Like background, foreground texture can be described by a per-pixel Gaussian model [165], although position information is usually added to the feature vector. Histogram is also commonly used as a simple and adaptive method for extracting global statistics of foreground texture. Toyama et al. [155] applied histograms to reliable object regions detected by frame differencing and backprojected histogram statistics to the image to correct mislabeled foreground pixels.

Appearance modeling, including PCA [14, 13] and wavelets [80, 140], is a popular tool for modeling object texture. Black and Jepson [14] proposed a pyramid of eigenspace representations to allow coarse-to-fine matching for tracking. In order to deal with outliers, the eigenspace was built to represent a smaller set of canonical views and a parameterized transformation was estimated to convert an input image to the eigenspace. Jepson et al. [80] expressed image appearance in terms of responses from a wavelet-based steerable pyramid, which provided image features at different scales for coarse-to-fine differential motion estimation and stability assessment.

Shape modeling The most common shape model is based on contours [16]. The idea is to fit a spline curve to the contour of an object, thus expressing its shape by the control vectors of the spline curve, or by lower dimensional shape space vectors mapped from the former via a linear transformation. The shape space describes the variation of the object shape and can

be constructed in various ways. The key-frame method builds the shape space from a linear combination of sample shapes; the random sampling approach assumes a Gaussian distribution of the shape; PCA can be employed either in the curve space or shape space; residual PCA operates on a key-frame based shape space that does not totally cover a certain data set, and fills in missing components by PCA, thus retaining the clear interpretation of key frames.

In order to obtain object contours, edge detection is often a necessary first step [174]. The Canny edge detector [30] is well known to give the best results but is computationally expensive. Contour grouping techniques [49, 50] are usually needed here to group edge features that belong to a common structure.

Motion modeling Dynamic models [165, 16, 67] such as first- and second-order *auto-regressive* (AR) processes are often used to describe the trajectory of a stochastic motion. The first-order AR process is the simplest model to deal with changes of position and shape, and can impose incremental constraints when global constraints are weak. However, they are not able to model motions with arbitrary directional changes of velocity, or oscillations. The second-order AR process is a natural extension and meets the requirements for both translational and oscillatory motions. Dynamic models can be learned by various algorithms such as the dynamic learning algorithm [16] and the *expectation maximization* (EM) algorithm [115].

PCA is another approach to representing motion [15, 142, 119]. Black et al. [15] modeled motion discontinuities and non-rigid motions using PCA on optical flows. Sidenbladh et al. [142] used a multivariate PCA to train a walking human motion model to reduce the dimensionality of a parameter space. Ormoneit et al. [119] proposed a functional PCA, which used the *fast Fourier transformation* (FFT) to deal with missing data and enforce motion smoothness, in order to model periodic human motion cycles.

Integrated modeling As mentioned in Section 2.3.1, most segmentation systems use more than one source of information for object modeling. Wren et al. [165] combined a skin-color model, 2D contour shape model, and a dynamic motion model for object tracking. Haritaoglu et al. [67] distinguished humans from other objects (e.g. cars) using both silhouette shape

and periodic motion cues, separated multiple overlapping objects using projection histograms computed from silhouettes, and identified objects after occlusion based on a *temporal texture template* by integrating appearance, shape and dynamic models. Taking advantage of the fact that the appearance and shape of vehicles remain unchanged in satellite images, Tao et al. [150] adopted constant appearance and shape models for vehicles and approximated their motion by a translation and rotation after the projective planar background motion is compensated. Kang et al. [87] employed a polar distribution of Gaussian models of color and shape to represent foreground objects, which had the advantage of being invariant to translation, rotation and scale.

Shape and motion can also be learned together by a HMM [125] for recognition purpose. Brand [24] trained a HMM to learn the shape and motion of a 3D human body from 2D silhouettes. Oliver et al. [118] used HMM and *coupled HMM* (CHMM) to model human behaviors and interactions. Nair and Clark [111] proposed a HMM for human activity recognition in order to detect anomalous activities for video surveillance. The HMM can be learned using the Baum-Welch method or the EM algorithm [125].

Key-frame extraction and video segmentation have recently been combined by Fan et al. [101, 144], where the extracted key frames were used to estimate statistical models for model-based object segmentation, and object segmentation results were used to further refine the initially extracted key frames. Spatial-temporal features used include YUV color, x-y spatial location, time, and intensity change over time. A feature selection process is applied, not to reduce feature dimension but rather, to extract video key frames such that the feature space overlap among major objects is minimized.

Object-specific modeling limits a segmentation system to specific targets. A new modeling process is required whenever a new object is involved, which may be time-consuming and impractical for online processing.

2.3.4 Tracking

Although object segmentation does not require tracking, they are often coupled, as segmentation is a prerequisite of tracking, and tracking is helpful for propagating good segmentation results through an image sequence. Feedback from motion tracking can also be incorporated into adaptive background modeling [151]. Therefore, it is beneficial to examine existing tracking approaches as well.

Tracking algorithms can be divided into two categories: one based on static images, the other based on temporal coherence in image sequences. PCA-based eigentracking [14, 13] is a typical method that considers each image in a video as isolated and static. Its detection and tracking mechanisms are essentially a single matching process; thus eigentracking does not require sophisticated feature detection and object segmentation techniques. However, due to the lack of temporal information during tracking, eigentracking processes images independently of each other and is computationally expensive in the long run.

The Kalman filter is the simplest tracking algorithm to incorporate temporal coherence of an image sequence [88, 165, 16] and has been integrated into many systems [118, 146, 47, 87]. It consists of three steps: First, given the tracking result from the previous frame, the algorithm *predicts* the shape and motion of an object in the current frame based on a shape and motion model; second, it *measures* or *observes* detected features of an object in the current frame; last, it *assimilates* the observation into the prediction using information weighting. With the help of an adaptive validation gate, the Kalman filter is able to adjust its search region and, hence, handle cluttered background, partial occlusion and agile motion. However, as it assumes a unimodal Gaussian distribution for an object, the Kalman filter is able to track only a single object and is not able to recover once it loses the target.

The particle filter or the CONDENSATION algorithm [16, 76, 12, 130, 51] is proposed to overcome the shortcomings of the Kalman filter. The particle filter assumes a multi-modal non-Gaussian distribution of an object and uses factored sampling to approximate its distribution. The particle filter is able to track multiple objects and recover from a loss of tracking. However,

using a high dimensional parameter space, the particle filter requires a large number of samples and it proves to be computationally expensive to provide a good approximation of the distribution. One solution to this problem is to reduce the dimension of the parameter space by some compression technique such as PCA. Another solution is to use the *hybrid Monte Carlo* (HMC) filter.

The idea of HMC [113, 35, 122, 174], also known as the *Markov chain Monte Carlo* (MCMC) filter, is to use fewer particles, where each particle produces a Markov chain to explore the parameter space until the chain converges to a posterior. Thus, the HMC explores the parameter space both in space and in time, which speeds up the searching process. In a study by Choo and Fleet [35], the HMC performed faster for problems of high dimensionality such as 10D or 28D, while the particle filter was faster for problems of low dimensionality such as 4D.

2.3.5 Layered motion segmentation

Layered motion segmentation decomposes an image sequence into a set of overlapping layers with each layer's motion described by a smooth optical flow field. The discontinuities in the description are attributed to object occlusions, mirroring the structure of the scene. This is a 2.5D representation because the segmentation result consists of motion layers as well as occlusion and depth ordering.

Layered motion originated from optical-flow based image motion analysis. The assumption made in many optical-flow algorithms is that motion at any point in an image can be represented as a single pattern component undergoing a simple translation; even complex motion appear as a uniform displacement when viewed through a sufficiently small window. This assumption fails for a number of situations, for example, transparent surfaces moving past one another, or at the boundary between two differently moving regions. Bergen et al. [9] addressed the problem by extending a coarse-to-fine incremental single motion estimator to a two-component motion model. The estimations of two motions are refined in an alternating iterative fashion.

Wang and Adelson's motion segmentation [160] estimated multiple affine motions within subregions of images using a multi-scale coarse-to-fine algorithm and determined coherent motion

regions based on the estimated affine models. The affine models and the regions were updated iteratively. Meanwhile, Jepson and Black [78] used a probabilistic mixture model on optical flow to represent multiple motions within an image patch explicitly using a simple modification of the EM algorithm for parameter estimation. Hsu et al. [73] presented a framework where a global parametric motion was estimated for an entire motion layer, and the motion at each point was corrected by residual flow, parametric estimation error, and residual intensity. Similar to the mixture model of Jepson and Black [78], Ayer and Sawhney [3] represented the current image by an additive mixture of a finite number of previous images that were warped by different motion models, and the parameters of these models were estimated by a modified EM algorithm. Weiss and Adelson [162] extended the above work [78, 3] by adding spatial constraints to the mixture formulations. A *Markov random field* (MRF) was employed to propagate neighborhood support and another modified EM algorithm was used for parameter estimation.

Due to its success in minimizing energy functions for stereo problems [22, 23, 92, 93], the graph cut approach has been adopted for motion layer extraction. Xiao and Shah [166] used two graph cut steps for motion layer extraction. First, seed regions, initially determined by two frame correspondences, are expanded and refined by a graph cut method and the resulting regions are merged into several initial layers according to motion similarity. Final motion layers are obtained by a multi-frame graph cut algorithm with the occlusion order constraint included in the energy function.

Layered motion segmentation methods almost all involve the computation of optical flow, which is time-consuming and impractical for real-time use. Besides, they do not exploit stereo information for depth extraction and are not able to provide precise depth information.

2.3.6 Stereo segmentation

The techniques for a single camera described above can be useful for segmentation and tracking with multiple cameras, yet new challenges arise, such as how to maintain the identity of an object across different views, and how to integrate information from different views, how to switch between cameras when sharing a large field of view.

Some researchers use multiple cameras mainly to provide a large field of view with little overlap of coverage between different views [29, 2]. In this context, the most important problems to be solved are matching and switching between different camera views. Cai and Aggarwal [29] proposed the use of geometry features—such as position and velocity—and intensity features for matching, and a camera selection scheme to choose one of the nearest cameras that had a high confidence match, and would image the object over the longest number of frames in the future. Atsushi et al. [2] projected a 3D ellipse model to 2D views to create various simulated images as estimations and compared them to actual background removed images so as to detect the 3D position of an object. Their view matching was based on 3D position and color.

In order to obtain more accurate 3D information, multiple cameras with overlapping fields-of-view have been used. Some researchers explored, without calibration, synchronized visual cues, such as the bottom points of a person moving on the ground [32, 89] or the edges of the field of view of a camera intersected by a person [89], to obtain the object correspondence in multiple views. Given multiple calibrated cameras, the matching can be done across different views more easily along epipolar lines. Mittal and Davis [107] back projected the centers of matching segments by stereo triangulation to obtain 3D points in a scene potentially corresponding to people. They then projected the 3D points to the ground plane for 3D tracking. Dockstader and Tekalp [47] used a Bayesian network for integrating back-projected 3D positions of features across multiple views. 2D segmentation and tracking were carried out based on regional optical flows in each view and a Kalman filter was adopted for maintaining the temporal smoothness of the 3D trajectory.

All of these methods used wide base-line camera arrangements, causing difficulties in stereo matching and uncertainty in object correspondence across multiple views. By exploring the potential of multiple calibrated small baseline cameras, some systems work from an extensive 3D reconstruction to object segmentation and tracking [77, 112], which relies on the good performance of a stereo algorithm. However, frame-by-frame stereo reconstruction is expensive

and so far unsuitable for real-time or interactive applications. Existing stereo matching algorithms [116, 131, 93, 149] also tend to be very sensitive to differences in camera response; obtaining sufficiently well-matched cameras may pose a real practical difficulty. Finally, the uniform or repetitive textures common in indoor scenes and video-augmented spaces constitute worst-case inputs for stereopsis, often leading to disappointing results.

The latest approach towards stereo segmentation is to combine the stereo matching and segmentation processes. Harville et al. [69] obtained depth information from a stereo camera and modeled the background using per-pixel, time-adaptive, Gaussian mixtures of combined color and depth. The background update rate was modulated based on scene activity and a depth-dependent color segmentation scheme was proposed. Similar to the idea of layered motion, the layered stereo approach [5, 154] models a scene as planar layers using 3D plane equations refined by per-pixel opacity and per-pixel residual depth relative to the plane. The multi-way cut segmentation approach [11, 99] reviewed in Section 2.2.3 extends this idea by representing the scene structure as a collection of smooth surface patches. An energy function is minimized by alternating between an image segmentation and a surface fitting process. Despite their good performance, the high computational cost remains a weakness of these methods.

Kolmogorov et al. [91] proposed both *layered dynamic programming* and *layered graph cut* for bi-layer segmentation. *Layered dynamic programming* solves stereo in an extended 6-state space that represents background and foreground layers and occluded regions. *Layered graph cut* does not solve stereo directly; instead, it segments an image into foreground and background layers based on a contrast-sensitive color model. Both methods yield comparably high accuracy in experimentation and have the potential for real-time application. However, it is noticeable that there is a substantial depth difference between background and foreground in the test images of these two methods. Although dynamic background such as people walking across the room was tested, foreground motion was restricted to a small depth range, close to the camera, which does not fit most real-world applications.

A few real-time stereo systems have been claimed in the past decade [85, 110, 55]. A close examination of these systems revealed that they give different definitions of "real-time". By using special-design dedicated processor boards and programmable hardware, the Video Rate Stereo Machine of Kanade et al. [85] generates depth images of 256×240 pixels at 30 frames/sec. The trinocular stereo system designed by Mulligan et al. [110] reconstructs disparity maps of foreground objects at 2-3 frames/sec at a resolution of 320×240 pixels and 64 disparities on a four-processor machine. The real-time segmentation system developed by Feldmann et al. [55] segments foreground objects from background in a color scene in real-time TV resolution by the support of multiple PCI based multi-processor boards containing four TriMedia TM 1300 on each board. Note that all these systems rely on the computational power of special hardware, which may not be affordable by many research labs. Moreover, they assume a static background for foreground/background separation and do not address the issue of dynamic environments.

Having looked at existing literature, it is clear that the object segmentation problem with rapid background texture changes has not been studied. The remainder of the thesis will present an implemented solution to this problem, starting with the work on camera calibration.

CHAPTER 3
Camera Calibration

3.1 Introduction

The proposed stereo-based object segmentation system uses a small baseline camera pair of fixed focal length. In order to establish the correspondence between the 3D scene and the 2D views of the cameras and so exploit the depth information of a scene, estimating camera parameters is the critical first step[1] .

Although camera calibration is a mathematically well defined problem and numerous methods have been developed to provide solutions, these methods make different simplifying assumptions as to how they should be used in practice and which variables need to be measured. It was not clear how certain factors such as training data quantity, measurement error, and the choice of camera model influence calibration accuracy. Moreover, a major practical concern is the degree of effort involved in providing a given algorithm with the training data it needs to achieve a required accuracy.

In order to answer the questions above, an empirical study was conducted, using three publicly available techniques, through extensive experimentation with separate training and test data, to investigate the impact of noise, either in world or pixel coordinates, and training data quantity, on calibration accuracy. A detailed comparison of various models was also included to determine the relative importance of different distortion components.

This chapter presents some useful observations from the study, in the hope of first providing a quick answer to researchers who need an immediate camera calibration solution; and second,

[1] The material in this chapter was originally published as Machine Vision and Applications 2006 Volume 17 pages 51-67, "An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques", Wei Sun and Jeremy R. Cooperstock,

giving some insight into practical difficulties one may encounter in a real environment, and which factors to keep in mind when looking for a suitable calibration technique; and finally, offering an introduction to the field for those newly embarking upon calibration related research.

3.2 Calibration methods for experimentation

The calibration methods chosen for experimentation were developed by Tsai [158], Heikkilä [71] and Zhang [172]. An important reason behind this choice is that the source code for the three methods is publicly available, well developed, well tested, and therefore provides a fair comparison. In fact, only open source algorithms are serious candidates for a study of this nature, because they are open to study and readily integrated into a larger system.

Tsai's method represents a conventional approach that is based on the *radial alignment constraint* (RAC) and requires accurate 3D coordinate measurement with respect to a fixed reference. Among the conventional calibration methods surveyed by Salvi [132], including those developed by Hall [65], Faugeras-Toscani [54], and Weng [163], Tsai's was reported to exhibit the best performance. His method has been widely used in multi-camera applications [86, 127]. Heikkilä's method, also world-reference based, although not included in that survey, employs the more general *direct linear transformation* (DLT) technique by making use of the prior knowledge of intrinsic parameters. It also incorporates a more complete camera model for lens distortions. Zhang's method is (essentially) a different special case of Heikkilä's formulation. It combines the benefits of world-reference based and auto-calibration approaches, which enables the linear estimation of all supposedly constant intrinsic parameters. This method is flexible in that either the camera or the planar training pattern can be moved freely, and the calibration procedure is easily repeatable without redoing any measurements. These three methods of course form part of a large space of potential algorithms and can each be generalized in a variety of ways. Researchers' names have been applied to particular forms of their algorithms as embodied in the published code.

3.2.1 Camera model

Tsai's, Heikkilä's and Zhang's methods all use the pinhole projective model to map 3D scenes to the 2D image plane. Despite different formulations for lens distortion, the mapping between world and image points proceeds through the same four transformations, from world coordinates (X_w, Y_w, Z_w), via camera coordinates (X_c, Y_c, Z_c), undistorted image coordinates (x_u, y_u), and distorted image coordinates (x_d, y_d), to pixel coordinates (x_p, y_p), as shown in Table 3–1.

The transformation between (X_w, Y_w, Z_w) and (X_c, Y_c, Z_c) is expressed by the *extrinsic parameters* $\mathbf{R}$ and $\mathbf{t}$, where $\mathbf{t} = [\, t_x \, t_y \, t_z \,]^T$ is a translation vector, and $\mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$ is a 3×3 rotation matrix which can also be expressed in terms of the roll-pitch-yaw angles, $\mathbf{r}_\theta = [\, \theta_x \, \theta_y \, \theta_z \,]^T$, as $\mathbf{R} = \begin{bmatrix} \cos\theta_y \cos\theta_z & \sin\theta_x \sin\theta_y \cos\theta_z - \cos\theta_x \sin\theta_z & \cos\theta_x \sin\theta_y \cos\theta_z + \sin\theta_x \sin\theta_z \\ \cos\theta_y \sin\theta_z & \sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z & \cos\theta_x \sin\theta_y \sin\theta_z - \sin\theta_x \cos\theta_z \\ -\sin\theta_y & \sin\theta_x \cos\theta_y & \cos\theta_x \cos\theta_y \end{bmatrix}$.

The other three transformations from (X_c, Y_c, Z_c) to (x_p, y_p) are expressed by the *intrinsic parameters*. In Tsai and Heikkilä's models, these are: the camera's principal point or image center in pixels , (c_x, c_y); the effective focal length, f; the image scale factor, s_x; the (supposedly known) center-to-center distances between adjacent pixels in x and y directions, d_x and d_y, or their inverses, D_x and D_y; the coefficients of Tsai's radial distortion, $k_1^{(T)}$; and Heikkilä's radial and decentering distortions, $k_1^{(H)}$, $k_2^{(H)}$, $p_1^{(H)}$ and $p_2^{(H)}$. In Zhang's model, the intrinsic parameters are: the image center, (c_x, c_y); the pixel focal lengths along the x and y axes, α and β; the radial distortion coefficients, $k_1^{(Z)}$ and $k_2^{(Z)}$; and the skew parameter, γ, describing the relative skewness of the x and y image axes of a camera. The focal lengths in the three models are related to each other as follows:

$$\alpha = f s_x / d_x = f s_x D_x, \quad \beta = f / d_y = f D_y. \tag{3.1}$$

Note that precise values are not needed for d_x, d_y and D_x, D_y in Tsai and Heikkilä's models as any error will be compensated for by the focal length, f, and the image scale factor, s_x, after calibration.

Table 3–1: Coordinate transformations in Tsai, Heikkilä and Zhang's camera models.

Tsai	**Heikkilä**	**Zhang**
$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t}$	$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t}$	$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t}$
$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix}$	$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix}$	$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix}$
$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \left(1 + k_1^{(T)} r^2\right) \begin{bmatrix} x_d \\ y_d \end{bmatrix}$ *where* $r = \sqrt{{x_d}^2 + {y_d}^2}$	$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \left(1 + k_1^{(H)} r^2 + k_2^{(H)} r^4\right) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1^{(H)} x_d y_d + p_2^{(H)} \left(r^2 + 2{x_d}^2\right) \\ p_1^{(H)} \left(r^2 + 2{y_d}^2\right) + 2p_2^{(H)} x_d y_d \end{bmatrix}$ *where* $r = \sqrt{{x_d}^2 + {y_d}^2}$	$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \left(1 + k_1^{(Z)} r^2 + k_2^{(Z)} r^4\right) \begin{bmatrix} x_u \\ y_u \end{bmatrix}$ *where* $r = \sqrt{{x_u}^2 + {y_u}^2}$
$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} s_x/d_x & 0 & c_x \\ 0 & 1/d_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$	$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} s_x D_x & 0 & c_x \\ 0 & D_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$	$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$

3.2.2 Calibration algorithms

All three calibration algorithms estimate an initial closed-form solution by solving part of an over-determined system of linear equations. The initial estimates then proceed through a non-linear optimization process, such as the standard Levenberg-Marquardt algorithm as implemented in Minpack [108], to minimize residual error. The objective function used in the optimization step is usually the error of distorted pixel coordinates, which will be described in detail in Sect. 3.2.3 Eq. (3.10). As the three algorithms differ mainly in their estimation of the closed-form solution, this section offers further detail as to how the initial solution is obtained in each algorithm.

In Tsai's algorithm, given the 3D world and 2D distorted image coordinates of $n \gg 7$ feature points, and assuming the camera's image center, (c_x, c_y), to be the center pixel of a captured image, a linear equation is formed based on the *radial alignment constraint* (RAC):

$$\begin{bmatrix} y_d X_w & y_d Y_w & y_d Z_w & y_d & -x_d X_w & -x_d Y_w & -x_d Z_w \end{bmatrix} \begin{bmatrix} t_y^{-1} s_x r_1 \\ t_y^{-1} s_x r_2 \\ t_y^{-1} s_x r_3 \\ t_y^{-1} s_x t_x \\ t_y^{-1} r_4 \\ t_y^{-1} r_5 \\ t_y^{-1} r_6 \end{bmatrix} = x_d. \tag{3.2}$$

With n such equations, an over-determined linear system can be established to solve for the unknowns, $s_x, t_x, t_y, r_1 \cdots r_6$. The third row of the rotation matrix, $r_7 \cdots r_9$, can be computed as the cross product of the first two rows. With the same n feature points, the focal length, f, and the last element of the translation vector, t_z, can be estimated from another over-determined system of n equations of the following form:

$$\begin{bmatrix} r_4 X_w + r_5 Y_w + r_6 Z_w + t_y & -d_y (y_p - c_y) \end{bmatrix} \begin{bmatrix} f \\ t_z \end{bmatrix} = d_y (y_p - c_y) (r_7 X_w + r_8 Y_w + r_9 Z_w), \tag{3.3}$$

where d_x and d_y are the distances between adjacent pixels in the x and y directions. The estimates of f and t_z and an assumption of zero for the radial distortion coefficient, $k_1^{(T)}$, then serve as an initial estimate for a Levenberg-Marquardt algorithm. Finally, all parameters derived from linear estimation and the image center, $(c_x,\ c_y)$, are optimized iteratively by the Levenberg-Marquardt algorithm to refine the solution.

Heikkilä's algorithm is based on the *direct linear transformation* (DLT) method [68], which is a general form of Tsai's algorithm. Ignoring nonlinear distortions, a linear projective transformation, $\mathbf{P}$, maps 3D world points, (X_w, Y_w, Z_w), to their corresponding pixel points, (x_p, y_p), up to a scale, μ:

$$\mu\,[x_p \;\; y_p \;\; 1]^T = \mathbf{P}\,[X_w \;\; Y_w \;\; Z_w \;\; 1]^T \tag{3.4}$$

where $\mathbf{P} = \mathbf{K}\,[\mathbf{R} \;\; \mathbf{t}]$, $\mathbf{R}$ and $\mathbf{t}$ are extrinsic parameters, and $\mathbf{K} = \begin{bmatrix} f s_x D_x & 0 & c_x \\ 0 & f D_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ determines the coordinate transformation between distorted image coordinates, (x_d, y_d), and pixel coordinates, (x_p, y_p), as shown in Table 3–1. Given $n \gg 5$ feature points, $\mathbf{P}$ is obtained from an over-determined system of n pairs of the following equation by eliminating μ:

$$\begin{bmatrix} X_w & Y_w & Z_w & 1 & 0 & 0 & 0 & 0 & -x_pX_w & -x_pY_w & -x_pZ_w & -x_p \\ 0 & 0 & 0 & 0 & X_w & Y_w & Z_w & 1 & -y_pX_w & -y_pY_w & -y_pZ_w & -y_p \end{bmatrix} \begin{bmatrix} \overline{\mathbf{p}}_\mathbf{1} \\ \overline{\mathbf{p}}_\mathbf{2} \\ \overline{\mathbf{p}}_\mathbf{3} \end{bmatrix} = 0, \tag{3.5}$$

where $\overline{\mathbf{p}}_\mathbf{i} (i = 1, 2, 3)$ is the ith row vector of $\mathbf{P}$. Setting f to an initial estimate, s_x to 1, and $(c_x,\ c_y)$ to the center pixel of an image, the extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ are computed by

$$\mathbf{R} = \mathbf{K}^{-1}\,[\mathbf{p_1}\ \mathbf{p_2}\ \mathbf{p_3}]\,, \quad \mathbf{t} = \mathbf{K}^{-1}\mathbf{p_4}, \tag{3.6}$$

where $\mathbf{p}_\mathbf{i} (i = 1, 2, 3, 4)$ is the ith column vector of $\mathbf{P}$. The Levenberg-Marquardt algorithm is then applied to find exact values of the intrinsic parameters including the distortion coefficients, $k_1^{(H)}$, $k_2^{(H)}$, $p_1^{(H)}$, $p_2^{(H)}$.

In Zhang's calibration process, a planar calibration pattern is presented to a camera in various orientations and is always assumed to be on $Z_w = 0$ of a changing world coordinate system. The calibration algorithm starts similarly to the DLT method except that the projective transformation, $\mathbf{P}$, in Eq. (3.4) now degenerates to a 3×3 homography, $\mathbf{H}$:

$$\mu \, [\, x_p \, y_p \, 1 \,]^T = \mathbf{H} \, [\, X_w \, Y_w \, 1 \,]^T, \tag{3.7}$$

where $\mathbf{H} = \mathbf{K} \, [\, \mathbf{r_1} \; \mathbf{r_2} \; \mathbf{t} \,]$, $\mathbf{K} = \begin{bmatrix} \alpha & \gamma & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix}$ determines the coordinate transformation between distorted image coordinates, (x_d, y_d), and pixel coordinates, (x_p, y_p), as shown in Table 3–1, and $\mathbf{r_i} (i = 1, 2, 3)$ is the ith column vector of the rotation matrix, $\mathbf{R}$. Hence, for each view of the pattern, a homography, $\mathbf{H}$, is estimated from n feature points on the pattern as in Eq. (3.5) and refined by the Levenberg-Marquardt algorithm. Based on the constraint that the image of the two circular points must lie on the image of the absolute conic, the proof of which can be found elsewhere [68], a linear equation pair is formed as follows:

$$\begin{bmatrix} \mathbf{v_{12}}^T \\ \left(\mathbf{v_{11}}^T - \mathbf{v_{22}}^T \right)^T \end{bmatrix} \mathbf{b} = 0, \tag{3.8}$$

where $\mathbf{v_{ij}} = \begin{bmatrix} h_{i1}h_{j1} & h_{i1}h_{j2} + h_{i2}h_{j1} & h_{i2}h_{j2} & h_{i3}h_{j1} + h_{i1}h_{j3} & h_{i3}h_{j2} + h_{i2}h_{j3} & h_{i3}h_{j3} \end{bmatrix}^T$, $[h_{i1} \, h_{i2} \, h_{i3}]^T$ is the ith column vector of a homography, $\mathbf{H}$, and $\mathbf{b} = [\, b_{11} \, b_{12} \, b_{22} \, b_{13} \, b_{23} \, b_{33} \,]^T$ is a 6D vector representation of the symmetric matrix

$$\mathbf{B} = \lambda \, \mathbf{K}^{-T} \, \mathbf{K}^{-1} = \lambda \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} = \lambda \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2 \beta} & \frac{c_y \gamma - c_x \beta}{\alpha^2 \beta} \\ -\frac{\gamma}{\alpha^2 \beta} & \frac{\gamma^2}{\alpha^2 \beta^2} + \frac{1}{\beta^2} & -\frac{\gamma (c_y \gamma - c_x \beta)}{\alpha^2 \beta^2} - \frac{c_y}{\beta^2} \\ \frac{c_y \gamma - c_x \beta}{\alpha^2 \beta} & -\frac{\gamma (c_y \gamma - c_x \beta)}{\alpha^2 \beta^2} - \frac{c_y}{\beta^2} & \frac{(c_y \gamma - c_x \beta)^2}{\alpha^2 \beta^2} + \frac{c_y^2}{\beta^2} + 1 \end{bmatrix}, \tag{3.9}$$

where λ is an arbitrary scale factor. Given $m \geq 3$ views of the pattern, *i.e.*, the m homographies obtained above, an over-determined system of m pairs of the above equation can be established to obtain the vector, $\mathbf{b}$, from which the intrinsic parameters can be extracted. The extrinsic

parameters with respect to each orientation of the pattern plane are computed by Eq. (3.6). Finally, the Levenberg-Marquardt algorithm optimizes the result while taking into account the radial distortions, $k_1^{(Z)}, k_2^{(Z)}$.

3.2.3 Evaluation of calibration accuracy

In the experiments, a camera is calibrated with Tsai, Heikkilä and Zhang's methods using a set of training data, and then validated with a neutral test set covering a wide distance range.[2] Some well developed, freely available calibration toolboxes are used as implementations of the three algorithms in order to achieve a fair comparison: Reg Willson's code [164] for Tsai's algorithm, Janne Heikkilä's toolbox [70] for Heikkilä's algorithm, and Jean-Yves Bouguet's toolbox [18] for Zhang's algorithm. For evaluating both training and testing accuracies, four of the most frequently used methods [132] were adopted based on their applicability to the single camera calibration case.

The *error of distorted pixel coordinates*, E_d, is measured by computing the discrepancy between estimated pixel coordinates, $(\hat{x}_{pi}, \hat{y}_{pi})$, projected from measured world coordinates by the camera model with lens distortions, and observed pixel coordinates, (x_{pi}, y_{pi}), obtained from captured images:

$$E_d = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\hat{x}_{pi} - x_{pi})^2 + (\hat{y}_{pi} - y_{pi})^2}, \tag{3.10}$$

where n is the number of feature points.

The *error of undistorted pixel coordinates*, E_u, is measured by computing the discrepancy between estimated undistorted pixel coordinates, $(\hat{x}_{upi}, \hat{y}_{upi})$, projected from measured world coordinates without lens distortions, and observed undistorted pixel coordinates, (x_{upi}, y_{upi}), computed by removing distortions from observed pixel coordinates:

$$E_u = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\hat{x}_{upi} - x_{upi})^2 + (\hat{y}_{upi} - y_{upi})^2}. \tag{3.11}$$

[2] In real data experiments, training sets are acquired separately for each of the three methods according to their varying requirements.

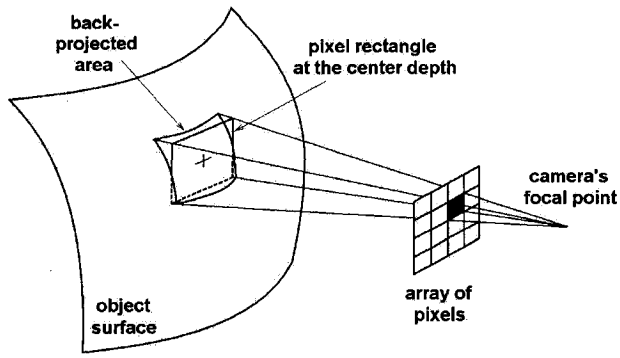


Figure 3–1: Back-projection of a pixel to the object surface. The back-projected area on the object surface is represented by a pixel rectangle at the center depth. The error between the back-projected area and the pixel rectangle is measured to assess the calibration accuracy. Based on Fig. 6 of Weng et al. 1992. [163]

The *distance with respect to the optical ray*, E_o, is measured between 3D points in camera coordinates, (X_{ci}, Y_{ci}, Z_{ci}), and the optical rays back-projected from the corresponding undistorted image points on the camera image plane, (x_{ui}, y_{ui}). For Tsai and Heikkilä's models, E_o is expressed as:

$$\begin{aligned} E_o^{(T\_H)} &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\sqrt{(X_{ci}-x_{ui}\cdot t)^2+(Y_{ci}-y_{ui}\cdot t)^2+(Z_{ci}-f\cdot t)^2}, \\ t &= \left(X_{ci}x_{ui}+Y_{ci}y_{ui}+Z_{ci}f\right)/(x_{ui}^2+y_{ui}^2+f^2)\ ; \end{aligned} \tag{3.12}$$

and for Zhang's model,

$$\begin{aligned} E_o^{(Z)} &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\sqrt{(X_{ci}-x_{ui}\cdot t)^2+(Y_{ci}-y_{ui}\cdot t)^2+(Z_{ci}-t)^2}, \\ t &= \left(X_{ci}x_{ui}+Y_{ci}y_{ui}+Z_{ci}\right)/(x_{ui}^2+y_{ui}^2+1)\ . \end{aligned} \tag{3.13}$$

These three measurements are intuitive but sensitive to digital image resolution, camera field of view, and object-to-camera distance. The *normalized calibration error* (NCE), proposed by Weng et al. [163], overcomes this sensitivity by normalizing the discrepancy between estimated and observed 3D points with respect to the area each back-projected pixel covers at a given distance from the camera. (See Fig. 3–1.) The NCE is calculated as follows:

$$E_n = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\left(\hat{X}_{ci}-X_{ci}\right)^2+\left(\hat{Y}_{ci}-Y_{ci}\right)^2}{Z_{ci}^2\left(\alpha^{-2}+\beta^{-2}\right)/12}\right]^{\frac{1}{2}}, \tag{3.14}$$

where $\left(\hat{X}_{ci}, \hat{Y}_{ci}, Z_{ci}\right)$ represent 3D camera coordinates as estimated by back-projection from 2D pixel coordinates to depth Z_{ci} and (X_{ci}, Y_{ci}, Z_{ci}) represent observed 3D camera coordinates computed from measured 3D world coordinates. In Tsai and Heikkilä's methods, the values of α and β can be calculated using Eq. (3.1). All four evaluation methods described in Sect. 3.2.3 were used and the results demonstrated very similar trends. Due to limited space, only the normalized calibration error (NCE) is plotted.

3.3 Evaluation on simulated data

The simulated camera had the following properties, chosen based on empirical data: center-to-center distances between adjacent pixels in x and y directions of $d_x = 1/D_x = 0.016$ mm and $d_y = 1/D_y = 0.010$ mm, an image scale factor $s_x = 1.5$, and an effective focal length $f = 8$ mm, resulting in pixel focal lengths of $\alpha = 750$ pixels and $\beta = 800$ pixels. A second-order radial distortion was simulated with the coefficients $k_1^{(Z)} = -0.32$ mm^{-2} and $k_2^{(Z)} = 0$ in Zhang's model, or equivalently, $k_1^{(T)} = k_1^{(H)} = 0.005709$ mm^{-2} and $k_2^{(H)} = p_1^{(H)} = p_2^{(H)} = 0$ in Tsai and Heikkilä's models. The skew parameter, γ, was set to 0. The image resolution was 512×512 pixels with the image center at $(c_x, c_y) = (264, 280)$ pixels.

The training points of all three methods, covering $30 - 55$ cm from the simulated camera, were obtained from the grid corners of a 20×20 cm simulated checkerboard patterns, placed at 16 different orientations in front of the virtual camera at a 45° angle with respect to the image plane.[3] The number of grid corners contained in a simulated pattern varies according to the requirements of each experiment.

The 4108 test points, covering a distance of $10 - 300$ cm, were generated by sampling a $3 \times 3 \times 3$ m cubic space at intervals of 10 cm in all three dimensions and keeping only points visible to a camera located at the center of one face of the cube.

[3] The number of orientations and the angle of the pattern plane were chosen based on Zhang [171], in which the best performance was reported with more than 10 orientations and an angle near 45°.

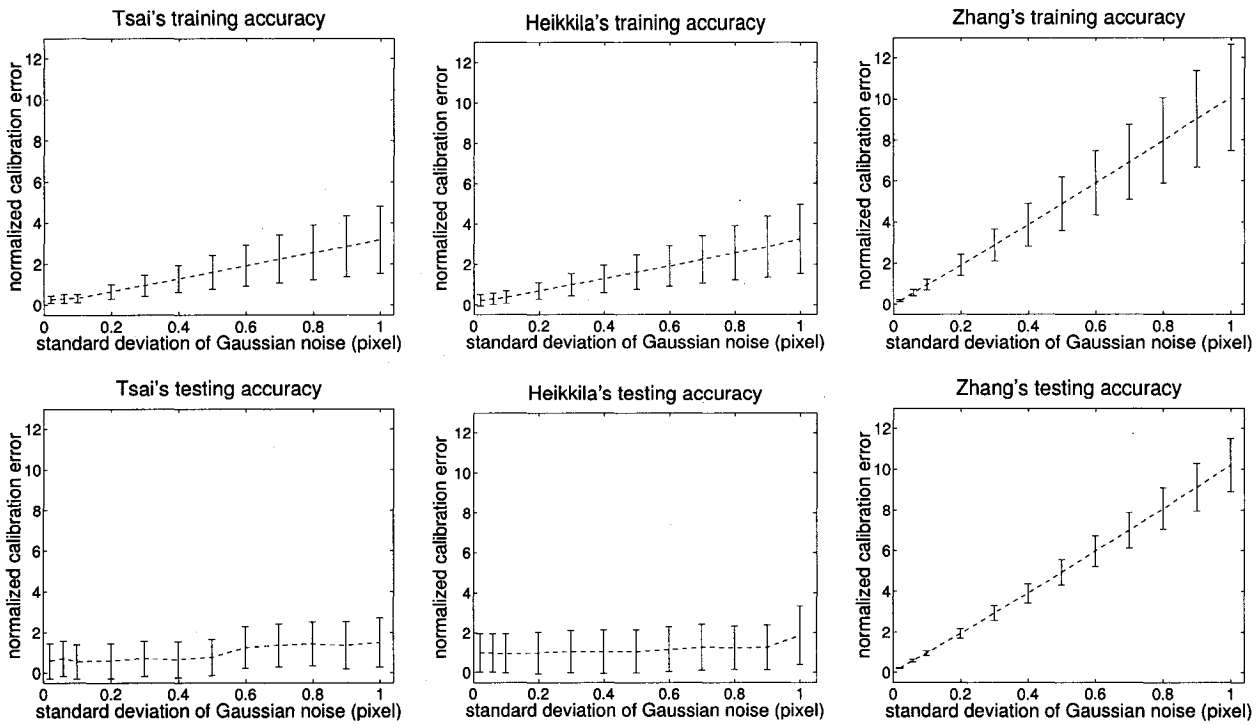


Figure 3–2: Effect of pixel coordinate noise on calibration accuracy. Top row: training errors vs. the amount of Gaussian noise (zero mean, σ of $0.02 - 1.0$ pix) added to training data pixel coordinates. Bottom row: testing errors vs. the amount of noise during training. No noise added to test data. Note that zero testing error was achieved when adding zero noise.

3.3.1 Effect of noise on calibration accuracy

A total of 16 views of a 10×10 checkerboard pattern were simulated to generate 1600 training points. Different levels of Gaussian noise were added to study its effect on calibration accuracy.

Calibration accuracy vs. pixel coordinate noise. Fig. 3–2 shows the decrease in training and testing accuracies of all tested methods as pixel noise increases. Zhang's algorithm was found to be more sensitive to pixel coordinate noise and hence more dependent on high-accuracy calibration feature detection than either Tsai or Heikkilä's.

Calibration accuracy vs. world coordinate noise. Fig. 3–3 illustrates the expected decrease of calibration accuracy as world coordinate noise increases. Zhang's calibration error was, again, the highest for the same noise range. While these results might at first seem discouraging for Zhang's method, it is important to note that unlike the other algorithms, which

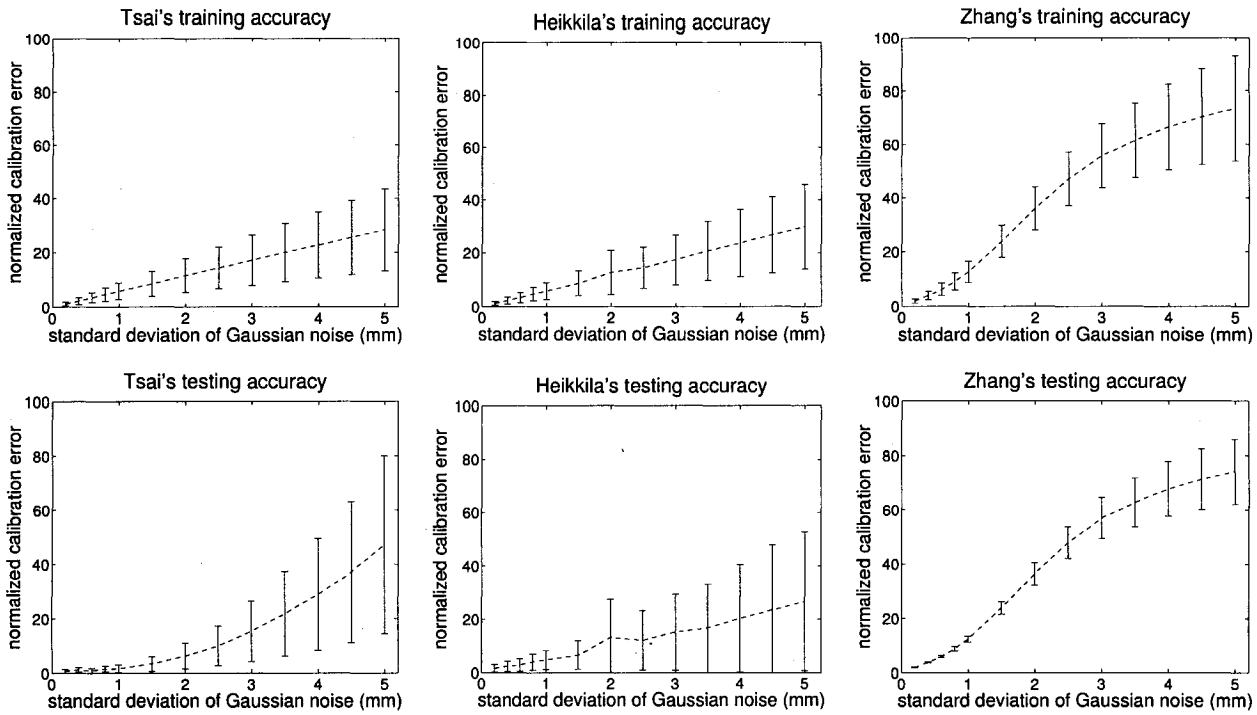


Figure 3–3: Effect of world coordinate noise on calibration accuracy. Top row: training errors vs. the amount of Gaussian noise (zero mean, σ of $0.2 - 5.0$ mm) added to 3D world coordinates of Tsai's and Heikkilä's training data and 2D world coordinates of Zhang's training data, as Zhang's algorithm assumes all feature points fall on the $Z_w = 0$ plane. Bottom row: testing errors vs. the amount of noise during training. No noise added to test data.

require absolute coordinate measurement with respect to a fixed world reference, Zhang's relative world coordinate measurement allows for a minimal equipment requirement, which, in practice, can simply be a laser printed checkerboard pattern on a letter sized sheet. As a result, most setups can easily achieve measurement noise levels of $\sigma < 0.5$ mm, obtaining a reasonably high calibration accuracy, as shown in the last column of Fig. 3–3. In contrast, Tsai and Heikkilä's world coordinate measurements are inherently prone to higher measurement error; the fact that their testing errors increased significantly when $\sigma > 2$ mm poses a strong constraint on real-world setups for accurate measurement. Although largely similar, Heikkilä's algorithm performed better than Tsai's at higher noise level, but slightly worse at lower noise levels. This may be due to Heikkilä's use of fixed empirical values to initialize the linear estimation of intrinsic parameters, which, while more robust to measurement errors, failed to exploit the potential of accurate measurements.

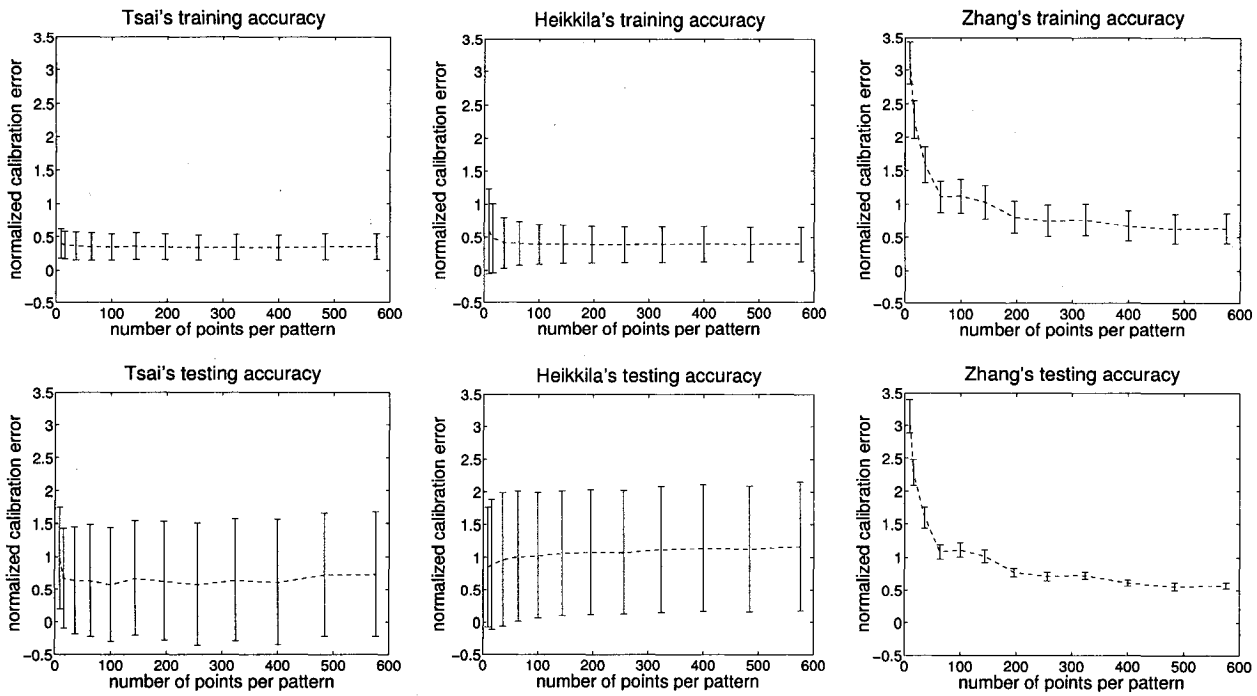


Figure 3–4: Training (top row) and testing (bottom row) errors vs. the number of training points per pattern. Training data generated from 16 views of checkerboard patterns containing between 3×3 and 24×24 grid corners. Gaussian noise of zero mean, $\sigma = 0.1$ pix added to training data pixel coordinates. No noise added to test data.

3.3.2 Effect of training data quantity on accuracy

As the effect of the number of orientations and the angle of the pattern plane on calibration accuracy has been studied by Zhang [171],[4] this section explores the effect of the number of feature points per pattern on calibration accuracy. In the absence of noise, a small number of training points can yield 100% accuracy. As some existing corner detection algorithms claim an accuracy of 0.1 pixels, Gaussian noise of zero mean and $\sigma = 0.1$ pixels was added to the pixel coordinates of training data. The average results of 10 trials are illustrated in Fig. 3–4.

Since each checkerboard pattern was viewed at 16 different orientations, a pattern of 3×3 grid corners could produce 144 training points, sufficient for Tsai's algorithm to achieve reasonable accuracy; however, the calibration error stabilized further when more than 256 training

[4] See the footnote on page 47.

points were used. Although more robust with limited training data quantities, Heikkilä's results demonstrated a slightly inferior performance to Tsai's, which might be explained by the low noise level in training data as suggested in the previous section. Zhang's calibration error was again higher than those of the other two methods with small training sets, as the former is more sensitive to noise. However, increasing the number of training points per pattern alleviates this sensitivity, resulting in similar accuracies to Tsai's algorithm when employing more than 200 training points per pattern. As noticeable in Fig. 3–4, testing errors exhibit higher standard deviation with Tsai and Heikkilä's algorithms than with Zhang's. This is likely due to the fact that the former two treated each feature point independently whereas the latter took advantage of the coplanar constraint between feature points of each view, thus compensating for the inconsistencies of noisy training points.

3.3.3 Effect of distortion model on calibration accuracy

Radial and decentering distortions [143] are the most common distortions modeled in camera calibration and can be expressed as follows:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots\right) \begin{bmatrix} x_u \\ y_u \end{bmatrix} + \begin{bmatrix} 2p_1 x_u y_u + p_2\left(r^2 + 2x_u^2\right) \\ p_1\left(r^2 + 2y_u^2\right) + 2p_2 x_u y_u \end{bmatrix} \tag{3.15}$$

where $k_i (i = 1, 2, \ldots)$ and p_1, p_2 are radial and decentering distortion coefficients, and $r = \sqrt{x_u^2 + y_u^2}$.

In this experiment, five types of cameras were simulated, each corresponding to a different distortion characteristic that consists of the first n low-order radial distortion terms with or without the two decentering terms, Rn (n = 1, 2) and RnD2 (n = 1, 2, 3). The simulated coefficients, listed in Table 3–2, were chosen from empirical data. All the remaining camera parameters were the same as previously described. Each of the five simulated cameras was calibrated by Zhang's algorithm combined, in turn, with each of the five distortion models, Rn (n = 1, 2) and RnD2 (n = 1, 2, 3), with skew parameter set to zero.

Fig. 3–5 shows, for each simulated camera, the calibration error versus the distortion models used for calibration on a large low-noise training set. The results indicate that high calibration

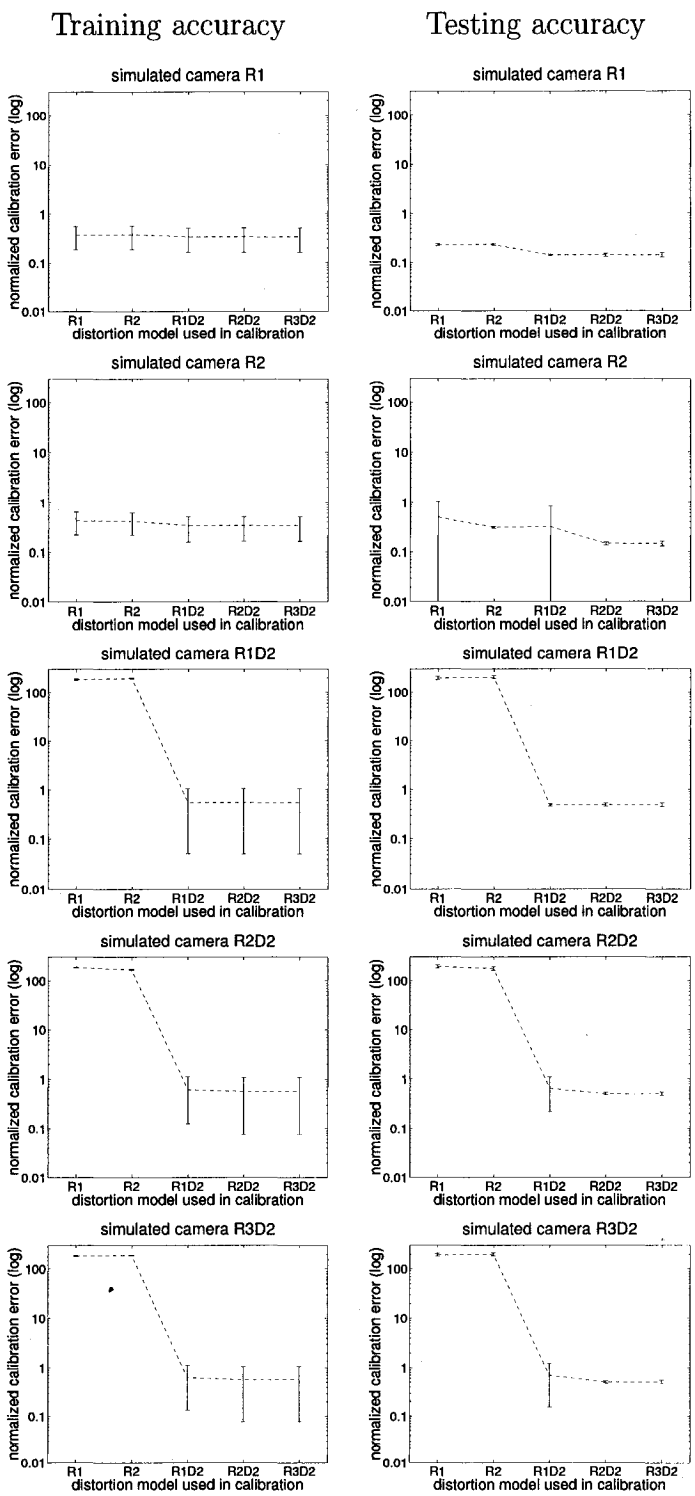


Figure 3–5: Training (left column) and testing (right column) errors vs. distortion model used in training. Trained from 16 views of 20×20 pattern with Gaussian noise of zero mean, $\sigma = 0.1$ pix added to pixel coordinates. No noise added to test data. Logarithmic scale used for y axis. Measured with Zhang's algorithm.

Table 3–2: Distortion coefficients of simulated cameras.

Distortion Coefficients	R1	R2	R1D2	R2D2	R3D2
k_1 (mm^{-2})	-0.3	-0.3	-0.3	-0.3	-0.3
k_2 (mm^{-4})	0	0.15	0	0.15	0.15
k_3 (mm^{-6})	0	0	0	0	0.1
p_1 (mm^{-2})	0	0	0.02	0.02	0.02
p_2 (mm^{-2})	0	0	0.015	0.015	0.015

Table 3–3: Simulated camera R2D2 and calibration results using different distortion models.

Camera Parameters	Simulated R2D2	Distortion Models Used in Calibration				
		R1	R2	R1D2	R2D2	R3D2
α (pix)	750	754.36	754.32	749.46	749.98	749.98
β (pix)	800	802.98	802.88	799.51	799.98	799.98
c_x (pix)	264	220.06	224.67	263.80	263.82	263.82
c_y (pix)	280	218.50	217.23	279.94	279.95	279.95
k_1 (mm^{-2})	-0.3	-0.1422	-0.1019	-0.2812	-0.3005	-0.3001
k_2 (mm^{-4})	0.15	–	-0.1460	–	0.1538	0.1472
k_3 (mm^{-6})	0	–	–	–	–	0.03055
p_1 (mm^{-2})	0.02	–	–	0.01995	0.01996	0.01996
p_2 (mm^{-2})	0.015	–	–	0.01497	0.01497	0.01497
Normalized Calib. Error	**training**	188.25	166.44	0.6179	0.5699	0.5701
	testing	196.14	174.20	0.6537	0.5102	0.5113

accuracy is obtained provided that the distortion model assumed in calibration includes all the distortion components of the camera, although the sixth order radial term does not benefit accuracy. Moreover, adding higher order radial terms affects the estimation of lower order terms, as can be observed in Table 3–3. This correlation between radial distortion components may, unfortunately, degrade calibration performance when only a limited amount of noisy training data is available. Fig. 3–6 illustrates the same experiment using noisy training data. The addition of higher order radial terms produced a higher error, especially with small training sets, as shown in the left column. Nonetheless, including the two decentering distortion components generally guaranteed a high calibration accuracy for a camera with unknown lens distortions.

3.4 Evaluation on real data

The real data experiments were carried out in our *Shared Reality Environment* (SRE), a laboratory space equipped with three vertical projection screens, each approximately 2.3×1.8 m

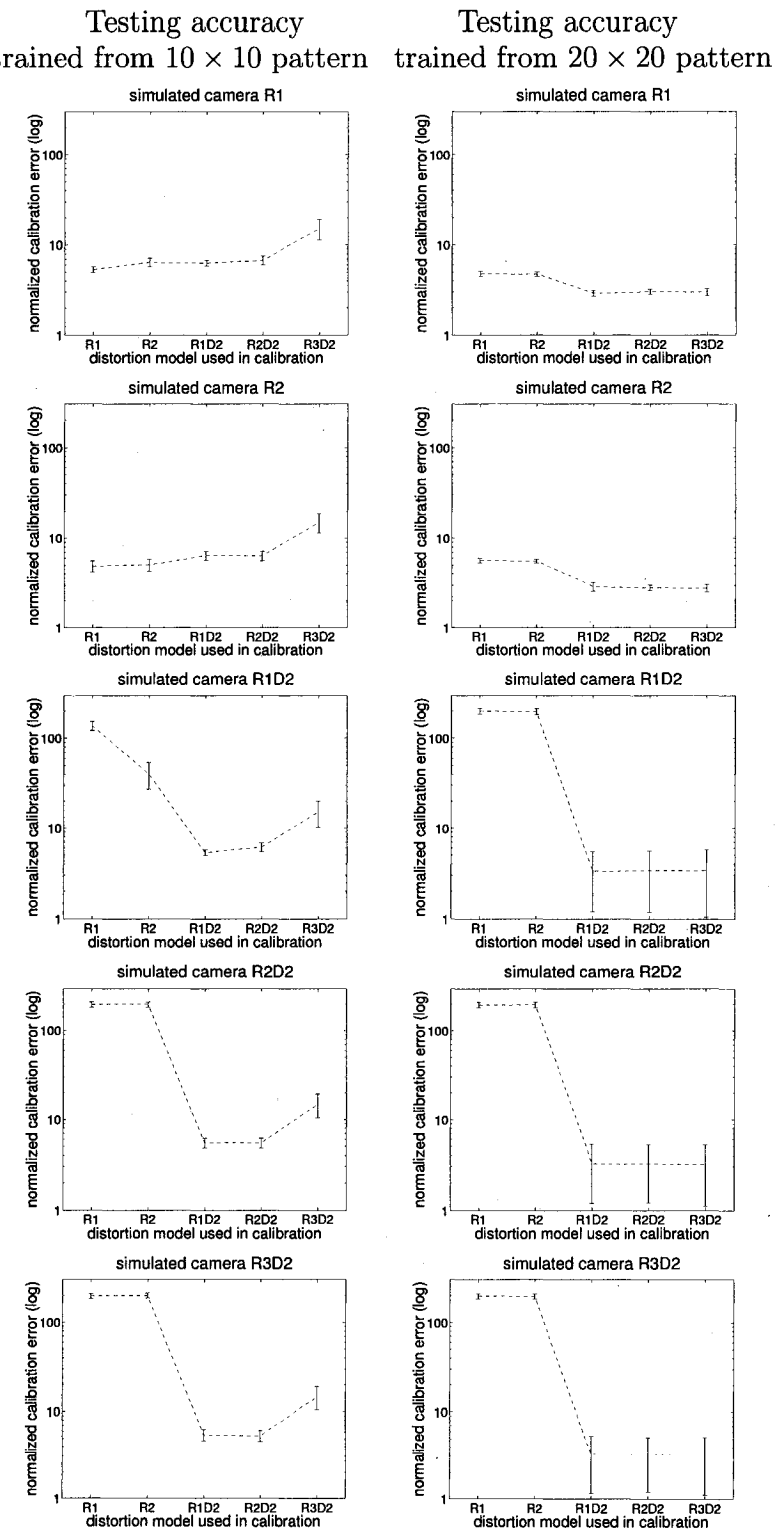


Figure 3–6: Testing errors vs. distortion model. Trained from 16 views of 10×10 (left column) and 20×20 (right column) patterns with Gaussian noise of zero mean, $\sigma = 0.5$ pix or 0.5 mm added to pixel and world coordinates. No noise added to test data. Logarithmic scale used for y axis. Measured with Zhang's algorithm.

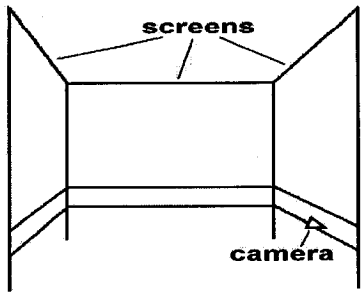


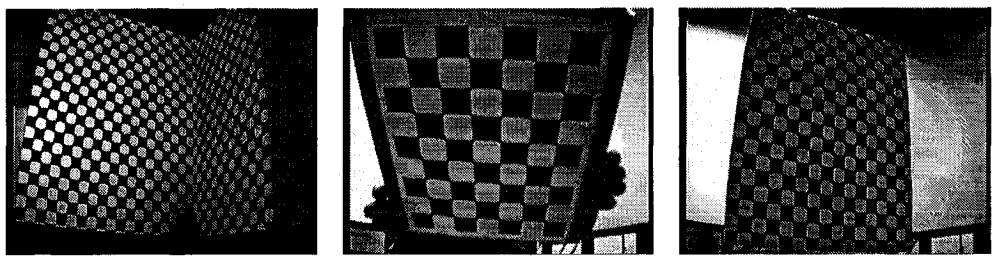
Figure 3–7: A node of the Shared Reality Environment.

and raised about 0.7 m above the ground, semi-enclosing a space of 5.4 m^2 as shown in Fig. 3–7. A 3Com U.S. Robotics BigPicture Camera with fixed focal length was placed along one screen at a height of 0.5 m, facing the other two adjacent screens. The image resolution was 640×480 pixels. To investigate the influence of experimental setup, two configurations, the *casual setup* and the *elaborate setup*, were studied.

3.4.1 Casual setup

The training data for Tsai and Heikkilä's calibration was obtained from 600 grid corners of checkerboard patterns of 8×8 cm squares, projected onto the two visible screens, as in Fig. 3–8(a).[5] Assuming that the projectors were accurately calibrated and the patterns were projected as rectangular shapes, the 3D world coordinates of the four pattern corners on each screen were measured according to a fixed world reference system referred to as SRE coordinates, and the remaining points were interpolated. Due to the calibration error between projectors and screens, the limited image resolution of the projected pattern, and the non-rigid material of the screens, verifying with a few interior points indicated an average interpolation error of 4.1 mm, equivalent to approximately 0.28% of the pattern size. This data set covered a distance of $200 - 325$ cm from the camera and will be referred to as Screen Data in the text below.

[5] Due to the difficulty of achieving 3D measurements of low errors on an arbitrarily positioned small checkerboard pattern, this experiment did not use 16 views of a hand-held checkerboard pattern for Tsai and Heikkilä's training data collection as in the simulation experiments, Sect. 3.3. Instead, large checkerboard patterns projected on big screens were used, which is likely to produce highly accurate measurements to the advantage of Tsai and Heikkilä's calibration.



(a) (b) (c)

Figure 3–8: A demonstration of *casual setup* for generating (a) Tsai and Heikkilä's training data, (b) Zhang's training data, and (c) test data.

The training data for Zhang's calibration was generated by printing checkerboard patterns of 8×6, 12×9, 15×14, and 20×20 grid corners onto letter sized sheets. Each was attached to a rigid card[6] and viewed at 16 different orientations at roughly 45° with respect to the camera image plane, as explained in Sect. 3.3 and demonstrated in Fig. 3–8(b). This produced four data sets of $768-6400$ points. The 2D relative world coordinates of these points were measured with respect to one of the four corners of each pattern, i.e. the origin of a changing world reference system. The four corners were measured first and the interior points interpolated due to the regularity of the printed pattern. Since the ruler was accurate only to 1mm, a maximum measurement error of 0.5mm was assumed, approximately 0.29% of the pattern size. These four data sets each covered a distance of $25-55$ cm from the camera and will be referred to as Board48, Board108, Board210, and Board400 Data respectively.

The test set for all three algorithms was created by moving a wooden board bearing a 100×85 cm printed pattern of 17×15 checks along a specially constructed rail at different locations within the SRE space, as pictured in Fig. 3–8(c), to provide 1872 accurately measured points in the SRE coordinates. This data set covered a distance of $100-265$ cm and will be referred to as Rail Data.

[6] According to Zhang's study [171], the effect of systematic non-planarity can be ignored in our experiments.

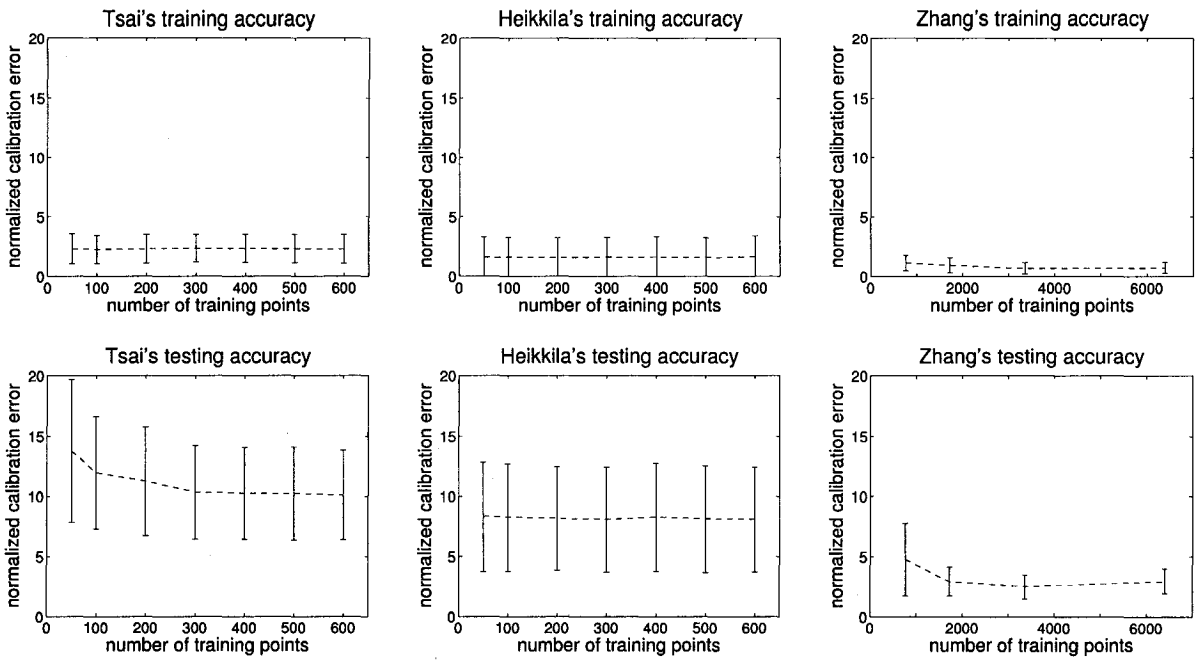


Figure 3–9: Training (top row) and testing (bottom row) errors vs. the number of training points in *casual setup*. Admittedly, more training points were used for Zhang's calibration than for Tsai and Heikkilä's. However, as indicated by the simulation results in Fig. 3–4, there was no accuracy improvement in Tsai and Heikkilä's results beyong 256 training samples. This is also evident in the present figure and Fig. 3–14 of the *elaborate setup*.

Effect of training data quantity. Tsai and Heikkilä's algorithms were trained using between 50 and 600 points, selected from Screen Data to be evenly distributed across the screens, and then tested on Rail Data. The average results of 10 trials for each quantity of training data are shown in Fig. 3–9. With Tsai's algorithm, no significant improvements in testing accuracy were observed beyond 300 training samples. Similarly to the simulation results, Heikkilä's algorithm produced better results than Tsai's with small training data quantities. The overall lower testing errors of Heikkilä's method compared to Tsai's suggested the presence of high noise in the training data, as discussed in Sect. 3.3.1.

Zhang's algorithm was trained separately on Board48, Board108, Board210, and Board400 Data, and tested on Rail Data. As Zhang's extrinsic parameters were calibrated with respect to a corner point of the calibration pattern, whose position in the SRE coordinates varied over different views, while the Rail Data were measured in the fixed SRE coordinates, the extrinsic

Table 3–4: The best calibration results obtained in *casual setup*.

Camera Parameters	**Tsai** (trained on Screen)	**Heikkilä** (trained on Screen)	**Zhang** (trained on Board210 with extrinsic recalib.)
Intrinsic Parameters	$c_x = 338.56$ pix $c_y = 240.34$ pix $f = 6.9333$ mm $s_x = 1.5575$ $d_x = 0.016$ mm (given) $d_y = 0.010$ mm (given) $k_1^{(T)} = 0.007121$ mm^{-2}	$c_x = 319.55$ pix $c_y = 263.12$ pix $f = 6.7617$ mm $s_x = 1.5504$ $D_x = 62.5$ mm^{-1} (given) $D_y = 100.0$ mm^{-1} (given) $k_1^{(H)} = 0.006477$ mm^{-2} $k_2^{(H)} = 0.000333$ mm^{-4} $p_1^{(H)} = 0.000627$ mm^{-2} $p_2^{(H)} = 0.001838$ mm^{-2}	$c_x = 332.39$ pix $c_y = 268.01$ pix $\alpha = 675.89$ pix $\beta = 695.15$ pix $\gamma = 0.000533$ $k_1^{(Z)} = -0.3230$ mm^{-2} $k_2^{(Z)} = 0.08667$ mm^{-4}
Extrinsic Parameters	$\mathbf{t} = \begin{bmatrix} -1982.9 \\ -1455.5 \\ 2324.9 \end{bmatrix}$ mm $\mathbf{r}_\theta = \begin{bmatrix} 153.18° \\ 19.44° \\ 91.25° \end{bmatrix}$	$\mathbf{t} = \begin{bmatrix} -1910.2 \\ -1535.9 \\ 2256.4 \end{bmatrix}$ mm $\mathbf{r}_\theta = \begin{bmatrix} 154.73° \\ 17.69° \\ 91.81° \end{bmatrix}$	$\mathbf{t} = \begin{bmatrix} -1963.5 \\ -1560.0 \\ 2269.7 \end{bmatrix}$ mm $\mathbf{r}_\theta = \begin{bmatrix} 153.07° \\ 17.33° \\ 91.45° \end{bmatrix}$

parameters needed to be recalibrated to the same coordinate system. The training data for this extrinsic recalibration was generated by placing a printed 12×15 checkerboard pattern at a location along the rail to produce feature points measured in SRE coordinates. After extrinsic recalibration, the testing accuracies on Rail Data were obtained, as shown in Fig. 3–9. Both training and testing accuracies increased with the number of training data per pattern with no significant improvement beyond 100 samples per pattern, which is consistent with the simulations. Recalling that the test data ($100 - 265$ cm from the camera) was obtained from a much further distance range than the training data ($25 - 55$ cm), one might expect large errors in the test results. However, despite this large discrepancy, Zhang's algorithm achieved impressive testing accuracies. This suggests that the parameters calibrated by Zhang's algorithm are scalable to test data over a larger range than represented in the training data.

Table 3–4 lists the best calibration results of each algorithm from Fig. 3–9, with the corresponding accuracy shown in Table 3–6, *casual setup*. Under our experimental conditions with the

interpolation measurement errors as described previously, Zhang's algorithm outperformed Tsai and Heikkilä's by approximately four and three times respectively, under all evaluation measures.

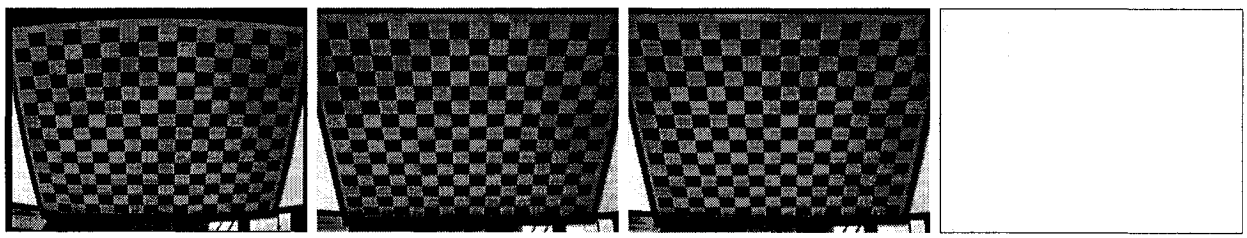
Effect of distortion model. The 3Com U.S. Robotics BigPicture Camera exhibits obvious lens distortions visible in the images of Fig. 3–8. The effect of distortion model on calibration accuracy was studied.

Skewness Although included in a linear transformation and not part of the actual distortion model, skewness should nevertheless be considered as a type of distortion. In Zhang's method, the skewness was estimated, as expressed by γ in Table 3–4, but was essentially zero. For comparison, camera parameters were calibrated by Zhang's algorithm on Board210 Data with γ either estimated or fixed at zero. The calibration accuracies are compared in Table 3–5, showing no improvement when estimating γ. This result can also be seen in Fig. 3–10.

Lens distortion Zhang's algorithm was integrated separately with each of the five distortion models described in Sect. 3.3.3 to calibrate camera parameters on Board210 Data. Calibration accuracies are displayed in Fig. 3–11. While all models achieved almost the same training

Table 3–5: Comparing calibration accuracies assuming skewness $\gamma \neq 0$ and $\gamma = 0$.

Accuracy	**Training Accuracy**		**Testing Accuracy**	
Evaluation	$\gamma \neq 0$	$\gamma = 0$	$\gamma \neq 0$	$\gamma = 0$
2D distorted error (pix)	0.2776	0.2782	1.0028	0.8959
2D undistorted error (pix)	0.2898	0.2905	1.0470	0.9395
distance from optical ray (mm)	0.1648	0.1653	2.7689	2.5153
Normalized Calibration Error	0.7099	0.7118	2.5602	2.3076



(a) (b) (c) (d)

Figure 3–10: Removing distortions from (a) original image using Zhang's model assuming (b) skewness $\gamma \neq 0$ and (c) $\gamma = 0$ with (d) their difference, i.e. $|(b) - (c)|$, which is barely visible. (White denotes zero difference.)

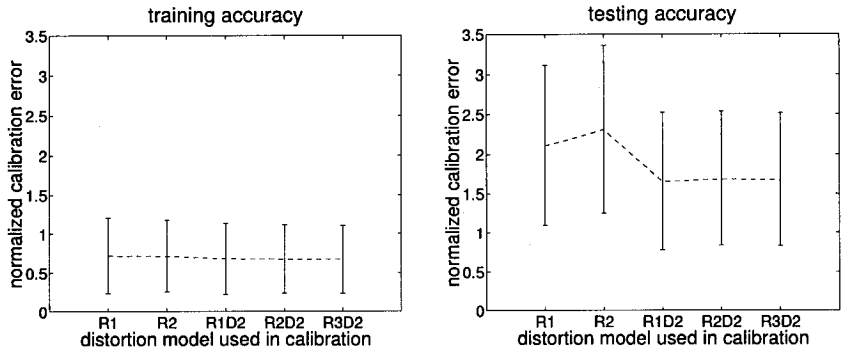


Figure 3–11: Training (left) and testing (right) errors vs. distortion model used in calibration. Trained from 16 views of a 15×14 checkerboard pattern. Measured with Zhang's algorithm.

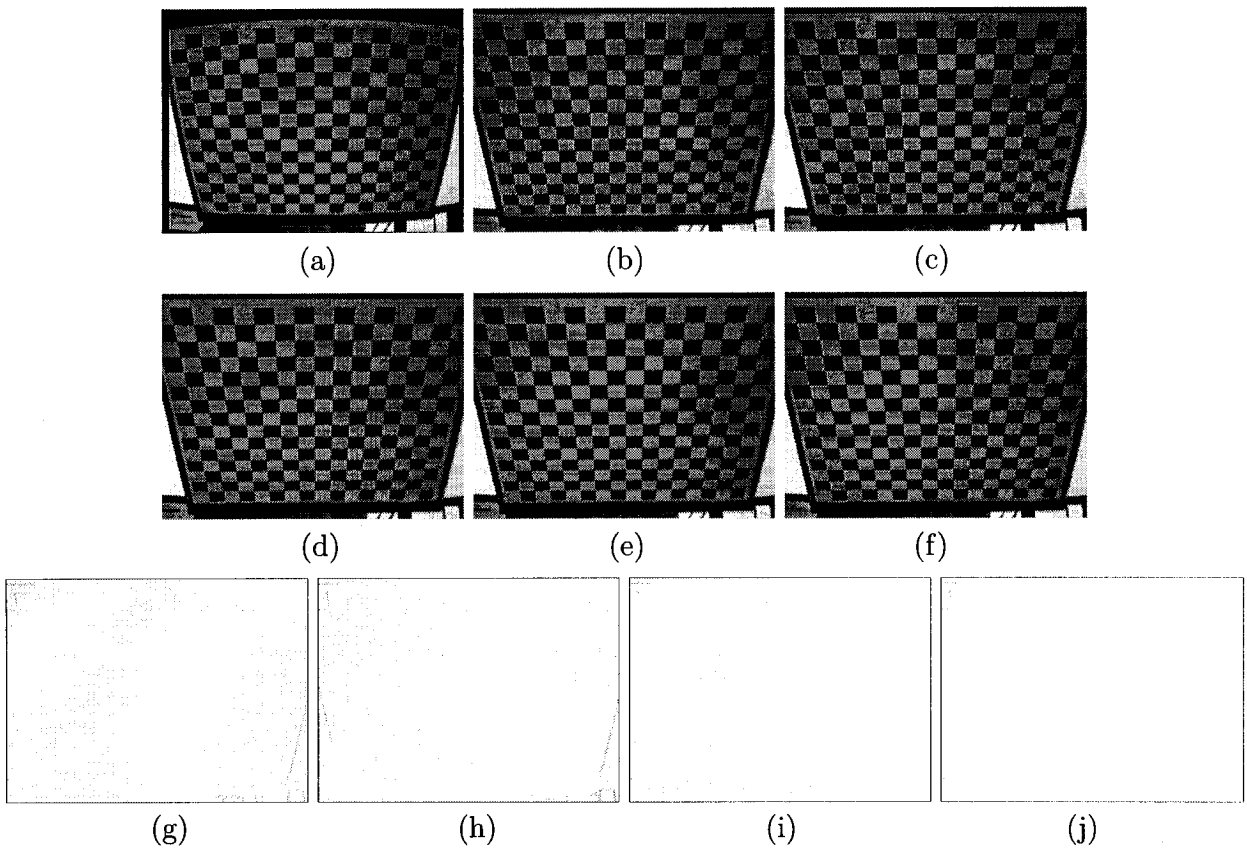


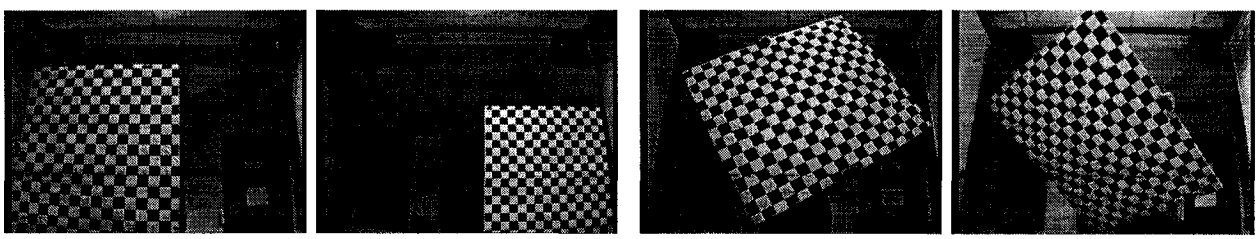
Figure 3–12: Removing lens distortions from (a) original image using distortion model (b) R1, (c) R2, (d) R1D2, (e) R2D2 and (f) R3D2. The differences of (b)(c)(d) and (e) with respect to (f) are shown in (g)(h)(i) and (j), respectively. (White denotes zero difference.)

accuracy, those considering decentering distortions performed marginally better in testing. As shown in Fig. 3–12, all five models successfully recovered the original image from distortions with little difference in the results.

3.4.2 Elaborate setup

To investigate how much improvement can be realized by increasing the measurement accuracy of training data and by reducing the discrepancy of distance coverage between training and test sets, the rail structure for obtaining the test data in Sect. 3.4.1 was used to generate a total of 3810 accurately measured feature points in an attempt to cover the volume of the camera's working space within the SRE. From this set, $50-2000$ evenly distributed points, covering $85-245$ cm from the camera, were selected as Tsai and Heikkilä's training data and the remaining points, covering the same range, were used as a neutral test set for all three algorithms, as demonstrated in Fig. 3–13(a). Zhang's training data was generated in the same manner as described in Sect. 3.4.1 but replacing the letter sized cardboard pattern with the 100×85 cm wooden board pattern of the rail structure, producing 16 views of $20-255$ planar points covering $95-225$ cm from the camera, as demonstrated in Fig. 3–13(b). The extrinsic parameters were then recalibrated with respect to the SRE coordinates by aligning this wooden board pattern with the projection screens at a known location on the rail.

The training and testing accuracy of all three algorithms are shown in Fig. 3–14 and Table 3–6, *elaborate setup*. Compared to the best results obtained in the *casual setup*, Tsai and Heikkilä's testing errors decreased by approximately 90% and 65% while Zhang's error decreased by only



(a) (b)

Figure 3–13: A demonstration of *elaborate setup* for generating (a) Tsai and Heikkilä's training data and test data, (b) Zhang's training data.

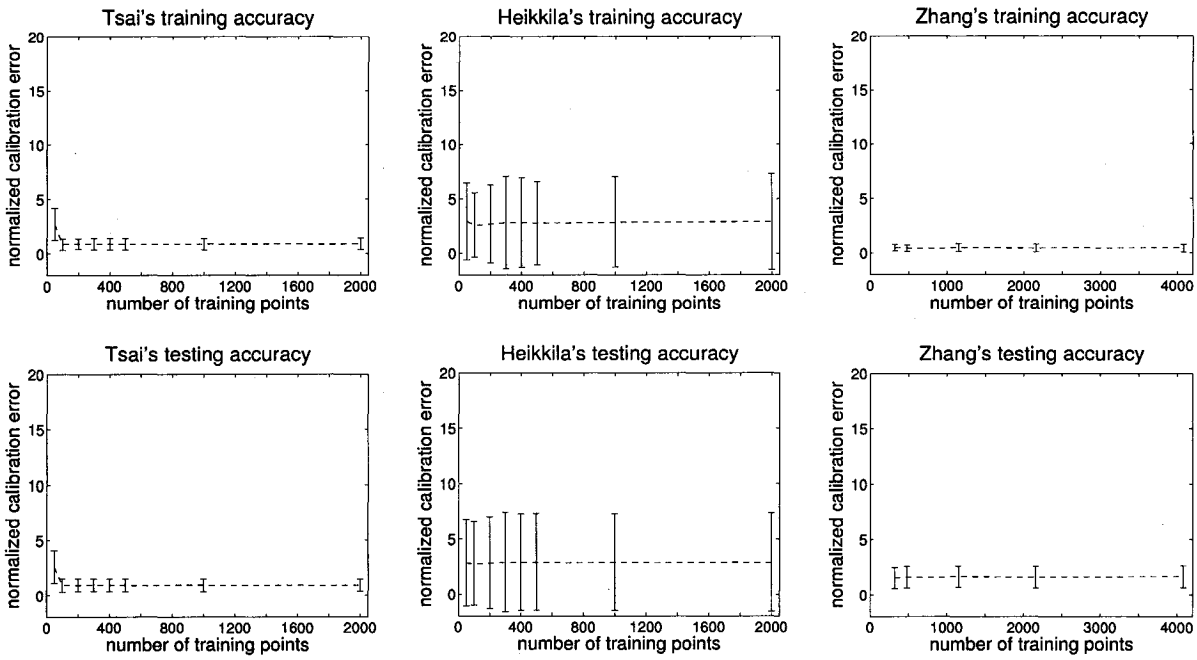


Figure 3–14: Training (top row) and testing (bottom row) errors vs. the number of training points in *elaborate setup*.

Table 3–6: Accuracy comparison of Tsai, Heikkilä and Zhang's calibration algorithms.

Setup	Accuracy Evaluation	Training Accuracy			Testing Accuracy		
		Ts	He	Zh	Ts	He	Zh
Casual	2D distorted (pix)	0.8843	0.4104	0.2776	3.9206	3.2311	1.0028
	2D undistorted (pix)	0.9410	0.6677	0.2898	4.0865	3.2949	1.0470
	3D distance (mm)	3.5752	2.4291	0.1648	9.9358	7.7337	2.7689
	Norm. Calib. Err.	2.3150	1.6397	0.7099	10.1184	8.0920	2.5602
Elaborate	2D distorted (pix)	0.3498	0.3480	0.1816	0.3445	0.3505	0.6301
	2D undistorted (pix)	0.3730	1.0765	0.1907	0.3701	1.1900	0.6856
	3D distance (mm)	0.9456	2.5730	0.4135	0.9244	2.8083	1.6926
	Norm. Calib. Err.	0.9141	2.6484	0.4667	0.9079	2.9267	1.6767

35% as, understandably, there was essentially no increase in training data accuracy of the latter. However, the distance change in training samples from the *casual setup* (25 – 55 cm) to the *elaborate setup* (95 – 225 cm) yielded a modest improvement in Zhang's results, as the test data (85 – 245 cm) was now closer in range to the latter.

3.5 Conclusion

An empirical study on camera calibration was carried out to investigate how such factors as noise level, training data quantity, and distortion model affect calibration accuracy. Three of the most popular and representative methods, developed by Tsai, Heikkilä and Zhang, were chosen for experimentation on both simulated and real data. Four commonly used criteria were applied to evaluate accuracy on separate training and test sets.

Results indicated that the conventional world-reference based approach, exemplified by Tsai's method, can achieve high accuracy when trained on data of low measurement error. However, this requires accurate 3D measurement, typically involving hundreds of samples with respect to a fixed reference system, which is prone to noise, and, as the experiments on actual data confirmed, yields a disappointing NCE of 10.1. After a careful and time-consuming setup and measurement process, the NCE was limited to approximately 0.9, indicating that the back-projected 3D error due to the camera parameters was lower, on average, than the error introduced by quantizing real world 3D coordinates to the level of individual image pixels [163]. However, the effort required to achieve this level of accuracy may well be inordinate for most researchers. Heikkilä's results demonstrated a similar trend despite its slightly greater robustness for small training data quantities and large measurement errors.

In contrast, the planar calibration approach, exemplified by Zhang's method, makes efficient use of world information by taking into account the planar constraint on the calibration pattern, and requires neither a laborious measuring task nor specialized equipment. With a hand-held pattern placed approximately 40cm from the camera, an NCE of around 2.6 was obtained, suggesting an acceptable calibration in which the residual error was almost negligible compared to the pixel quantization error discussed above [163]. Training with a pattern placed closer to the location of the test data yielded improved results, with NCE falling to 1.7.

Although the comparison between the three methods required that the extrinsic parameters obtained by Zhang's method be recalibrated with respect to the fixed SRE coordinates, this process may not be relevant for stereo or multi-camera calibration, in which a camera reference

system can be used instead of a world reference system. Moreover, the sensitivity of Zhang's algorithm to noise may be overcome by adding training points, simply by printing a checkerboard pattern containing more grid corners. In summary, these results demonstrate the flexibility of the multi-view planar approach and its suitability for calibration in dynamic environments.

The study also included a detailed comparison of distortion models to determine the relative importance of various coefficients given unknown lens distortion. The zero-skewness assumption made in many methods was confirmed as reasonable, at least for the cameras of average quality that were tested, and the second order term was found sufficient for modeling radial distortion [132]. Estimating the fourth order radial term may be desirable at low noise levels, although including the sixth order term can degrade calibration performance for small, noisy training sets. For a camera with unknown lens distortion, adding decentering components, in general, increases the likelihood of accurate calibration.

For calibrating the cameras of the proposed segmentation system, Zhang's algorithm was used, with a distortion model of two radial distortion terms and two tangential distortion terms, R2D2.

CHAPTER 4
Disparity Contours

As noted in Sect. 2.3.2, background subtraction is a simple and efficient method for segmenting foreground objects from a scene, provided that an accurate background model is available. Unfortunately, all background models developed so far [165, 145, 155, 118, 130, 64, 42] focus on the texture of the background, which impairs their adaptability to rapid changes in background texture and illumination.

This thesis presents a background subtraction approach based on the geometry instead of the texture of a background. With the help of a stereo camera pair, the background geometry is modeled by a *background disparity map*. Assuming that a given scene (where foreground objects may exist) contains only the background (*background hypothesis*), a mismatch between the scene and the background geometry model (*background hypothesis falsification*) can be expected to give clues to the location of foreground objects. It was discovered in the course of the study that *background hypothesis falsification* actually results in object boundary contours carrying foreground disparity information (*disparity contours*) rather than object areas. Such contours possess a number of features that can be exploited for object segmentation in a dynamically textured environment.

This chapter explains the idea of *background hypothesis falsification*, the resulting *disparity contours*, and the construction of *background disparity map*.

4.1 Background hypothesis falsification (BHF)

The 3D layout of a background scene as observed by a stereo camera pair can be represented by a *background disparity map* (BDM) that describes the relative displacement, or *disparity*, of pixels corresponding to the same background image point in each camera view. Since the two images have been rectified [60] during preprocessing, as described in Appendix A, pixels corresponding to the same scene point s differ only in x-coordinate, appearing at $(\mathbf{x}_L(s), \mathbf{y}(s))$

in V_L, the left view, and at $(\mathbf{x}_R(s), \mathbf{y}(s))$ in V_R, the right. The difference

$$\mathbf{x}_L(s) - \mathbf{x}_R(s) = \mathbf{d}_B(s), \tag{4.1}$$

which increases as distance to the camera decreases, is referred to as the *background disparity* at s. Thus, the BDM is defined to be the set

$$\text{BDM} = \{\langle \mathbf{x}_L(s), \mathbf{x}_R(s), \mathbf{y}(s) \rangle\}, \tag{4.2}$$

where s ranges over all background scene points visible to either camera and in the field of view of both. As the depth of a visible background point can be easily calculated from the disparity [157], the BDM actually encodes the 3D geometry of the background. Based on the observation that the background geometry is usually more stable over time than texture and illumination, the BDM can be useful as a valid geometric model over a long time even in the presence of lighting and texture change. This provides the basis for the following *background hypothesis falsification.*

Given the BDM for two cameras, each new pair of captured images are hypothesized to be of the background and a *view difference map* (VDM) is computed based on the stored correspondences, using a block-matching technique based on *sum of absolute differences* (SAD) in Eq. (2.2):

$$\text{VDM}_B(x_L, x_R, y) = \sum_{u,v} |V_L(x_L + u, y + v) - V_R(x_R + u, y + v)|, \quad \langle x_L, x_R, y \rangle \in \text{BDM}. \tag{4.3}$$

A pair of difference images, D_L and D_R, are constructed from the VDM for the points visible in the left and right views, as shown in Fig. 4–1. If the images are well synchronized, the difference cancels background texture, making the method applicable to dynamically textured scenes.

Assuming Lambertian reflectance or isotropic illumination, ideally $D_L(\mathbf{x}_L(s), \mathbf{y}(s)) = 0 = D_R(\mathbf{x}_R(s), \mathbf{y}(s))$ where a scene point s is truly part of the background, and yields a higher value if at least one of the pixels $V_L(x_L, y)$ and $V_R(x_R, y)$ belongs to a foreground object. As illustrated in Fig. 4–2, suppose $\langle a_L, a_R, y \rangle$, $\langle b_L, b_R, y \rangle$, $\langle c_L, c_R, y \rangle$ and $\langle d_L, d_R, y \rangle$ represent four entries on the y^{th} scanline in the BDM. At object boundaries, segments $[(a_L, y), (b_L, y)]$ and $[(c_L, y), (d_L, y)]$ in

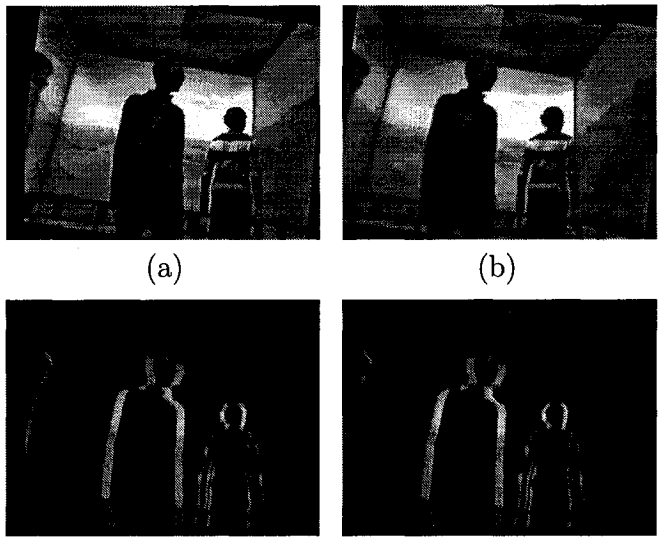


Figure 4–1: View differences under background hypothesis. Original (a) left view V_L and (b) right view V_R after image distortion removal and rectification. Corresponding projection (c) D_L and (d) D_R of computed view difference map. Note that high responses occur both at object boundaries and within objects of non-uniform textures.

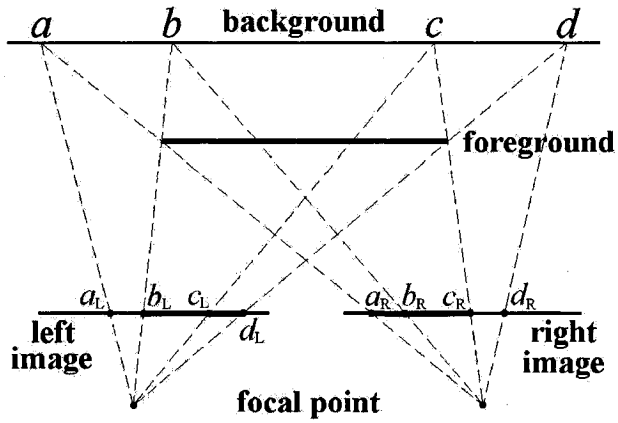


Figure 4–2: Background hypothesis falsification. Mismatch occurs both at object boundaries and interiors.

the left image are mismatched against segments $[(a_R, y), (b_R, y)]$ and $[(c_R, y), (d_R, y)]$ in the right image respectively. In object interiors, segment $[(b_L, y), (c_L, y)]$ in the left image is mismatched against segment $[(b_R, y), (c_R, y)]$ in the right. Thus, a mismatch value that is significantly different from zero leads to the falsification of the hypothesis that the BDM is an accurate local description

at the scene point. This is the essence of our geometric background subtraction. We wish to segment non-zero areas in the image that represent the foreground objects.

However, the reality is very different from our original expectation. The result of VDM depends on the visual difference between the background and foreground, between different foreground objects, and between internal points within a foreground object. It has been consistently found that mismatches at object boundaries due to foreground-background difference have exploitably higher intensity and more regular shape than those obtained in object interiors due to foreground self-correlation, as visible in Fig. 4–1. This motivates us to consider the possibility of extracting object boundary contours instead of object areas. We first examine more closely the characteristics of the mismatches at object boundaries.

4.2 Disparity contours

At object boundaries, stereo mismatch resulting from *background hypothesis falsification* forms contours, as illustrated in Fig. 4–3. Since disparity increases with proximity to the cameras, the width of the contour area in which background is mismatched against the foreground depends on how bad the assumption of background was, in terms of depth error.

Considering, without loss of generality, the left view, it is obvious from Fig. 4–2 that

$$\mathbf{x}_{\mathrm{L}}(a_{\mathrm{L}}) - \mathbf{x}_{\mathrm{R}}(a_{\mathrm{R}}) = \mathbf{d}_{\mathrm{B}}(a_{\mathrm{L}}) \tag{4.4}$$

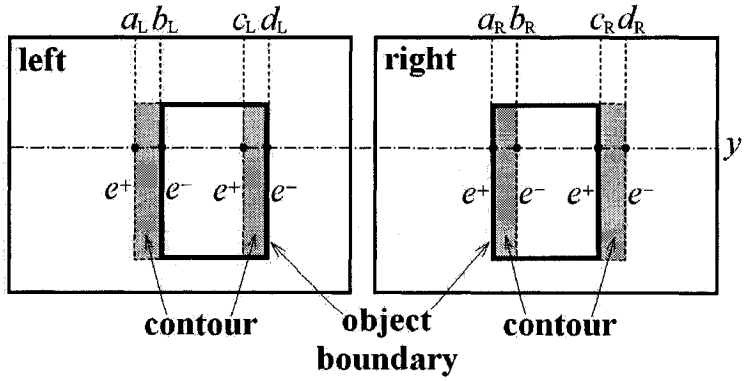


Figure 4–3: Disparity contours resulting from background hypothesis falsification. The width of the contour equals the relative disparity of an object with respect to the background.

according to the BDM, and

$$\mathbf{x}_{\mathrm{L}}(b_{\mathrm{L}}) - \mathbf{x}_{\mathrm{R}}(a_{\mathrm{R}}) = \mathbf{d}_{\mathrm{F}}(b_{\mathrm{L}}), \tag{4.5}$$

where $\mathbf{d}_{\mathrm{F}}(b_{\mathrm{L}})$ is the *foreground disparity* at b_{L}. Subtracting the two equations yields

$$\mathbf{x}_{\mathrm{L}}(b_{\mathrm{L}}) - \mathbf{x}_{\mathrm{L}}(a_{\mathrm{L}}) = \mathbf{d}_{\mathrm{F}}(b_{\mathrm{L}}) - \mathbf{d}_{\mathrm{B}}(a_{\mathrm{L}}). \tag{4.6}$$

Similarly,

$$\mathbf{x}_{\mathrm{L}}(d_{\mathrm{L}}) - \mathbf{x}_{\mathrm{L}}(c_{\mathrm{L}}) = \mathbf{d}_{\mathrm{F}}(d_{\mathrm{L}}) - \mathbf{d}_{\mathrm{B}}(c_{\mathrm{L}}). \tag{4.7}$$

This means that the lengths of the segments $[(a_{\mathrm{L}}, y), (b_{\mathrm{L}}, y)]$ and $[(c_{\mathrm{L}}, y), (d_{\mathrm{L}}, y)]$ are exactly the *differential disparities* between the foreground and background at the object boundaries, and encode depth. Combining such segments vertically in Fig. 4–3 will generate the depth-encoding contours of foreground objects, referred to as *disparity contours.* As the figure makes clear, the resulting contours lie at the left of object boundaries in the left image but at the right in the right image. There is thus no ambiguity as to the location of the object boundaries.

In order to maintain reasonably precise contour widths while aggregating neighborhood support during block matching, an 11×1 vertical stripe window is employed for Eq. (4.3). However, empirical study suggests that this choice of window height is not critical. Any value in the range of $7 \sim 15$ produced similar results.

4.3 Offline construction of background disparity map

The construction of the BDM is a well-understood problem. In the common case that the 3D geometry of the background is static, the BDM construction can be performed once, offline. This allows for the use of sophisticated stereo matching algorithms [116, 131, 93, 149, 99], for example, without impact on run time.

Unfortunately, such algorithms typically assume both well-matched cameras and non-uniformly textured backgrounds. The presence of large textureless regions in typical virtual reality environments such as that in Fig. 4–4(a)(b), and the varying intensity responses of low quality cameras, can produce poor results. Fig. 4–4(c) shows the disparity map computed from Kolmogorov and Zabih's graph cut algorithm [93] (software provided by Kolmogorov [90]) after

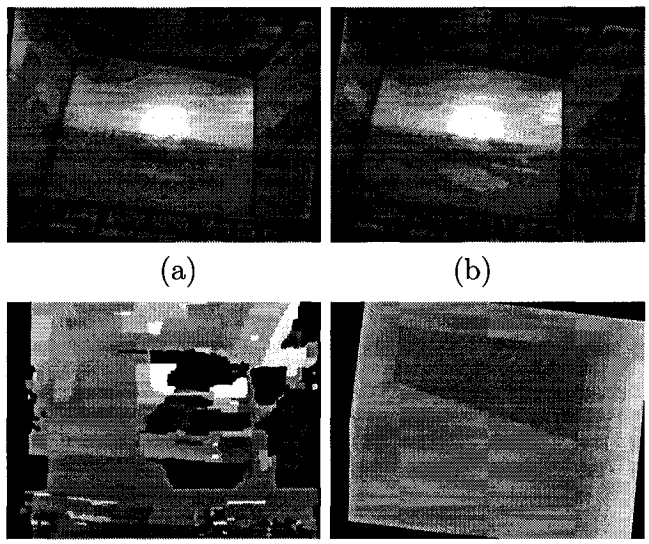


Figure 4–4: Background disparity map (BDM) construction. Original (a) left and (b) right background images after image distortion removal and rectification. (c) BDM by graph cut, left view. (d) BDM by ray tracing using 3D measurements of the background scene, left view.

a moderate amount of parameter tuning. Although their method was rated one of the best performers among stereo algorithms [137], its result on our background images is disappointing.

In the current implementation, the 3D geometry of the environment is measured manually by a steel tape measure and a ray tracing algorithm [57] is performed to obtain prior knowledge of the disparities. Fig. 4–4(d) shows the resulting disparity values from the BDM.

If more intrusive initialization procedures are acceptable, and the entire background scene can be actively illuminated, a structured light calibration method [170] should readily produce a BDM of comparable quality.

4.4 Disparity contour extraction and object segmentation

Disparity contour extraction aims to extract object boundary contours resulting from *background hypothesis falsification* while suppressing unwanted structures caused by excessive foreground object internal texture and camera calibration or synchronization error. Therefore, disparity contour extraction can be further divided into two steps, extraction and noise removal. Object segmentation then separates foreground objects from background based on extracted

contours. Experimentation with three different approaches, based on different strategies in contour extraction, noise removal or segmentation was performed with the results described in the following three chapters.

CHAPTER 5
Disparity Multihistogram Segmentation

The disparity multihistogram approach was the first attempt to segment objects from dynamically textured background. It employs a simple strategy in contour extraction and noise removal, and separates multiple objects at different depths via a multihistogram scheme. The system diagram is shown in Fig. 5–1.

5.1 Contour extraction

Contour extraction proceeds through edge detection, edge pairing, noise removal, and contour repair. Let D represent a difference image for either left or right view, as obtained from Eq. (4.3), an example of which is shown in Fig. 5–2(b). A simple $[-1\ 1]$ edge operator is applied to D to generate an edge image, E.

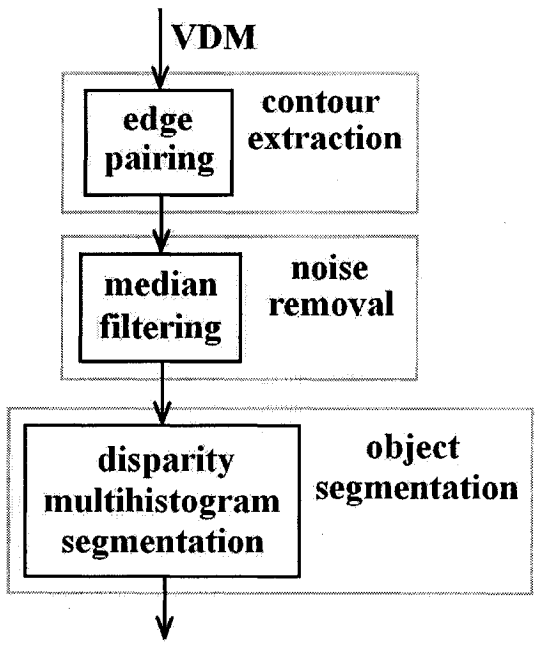


Figure 5–1: Disparity multihistogram object segmentation scheme.

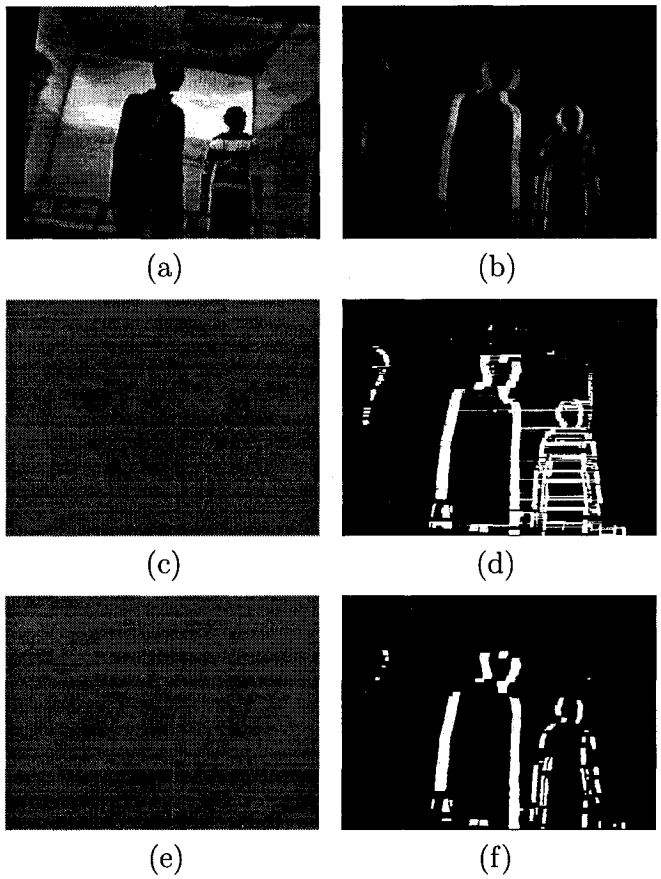


Figure 5–2: Contour extraction, left view. (a) Original image after image distortion removal and rectification. (b) Projection D of view difference map. (c) Contours C extracted by edge detection and pairing. (e) Result C_* after median filtering and contour repair. (d) and (f) are results in (c) and (e) displayed as binary line segments joining edge pairs.

Clearly, positive edges are obtained on the left side of a contour ridge in D and negative edges on the right side. Therefore, contours can be extracted by locating the corresponding positive and negative edge pairs, e^+ to the left of e^-, on the same scanline. They are required to satisfy the following two conditions:

$$\begin{aligned} &1.\ \mathrm{E}(e^+) \,/\, \max(\mathrm{E}) > \tau_{\mathrm{E}} \wedge \mathrm{E}(e^-) \,/\, \min(\mathrm{E}) > \tau_{\mathrm{E}}; \\ &2.\ |\mathrm{E}(e^+) + \mathrm{E}(e^-)| < \varepsilon_{\mathrm{E}}. \end{aligned} \tag{5.1}$$

Condition 1 excludes edge points below an empirically determined relative intensity threshold $\tau_{\mathrm{E}} = 0.06$, and condition 2 requires that e^+ and e^- have similar edge intensity in E. The line segments, c, between the paired positive and negative edge points are the desired contours, the

lengths of which, $|c|$, equal the differential disparity, $\mathbf{d}(c)$, between an object and the background. This information can be stored in a contour image, C, illustrated in Fig. 5–2(c).

$$|c| = \mathbf{d}(c) = \mathbf{x}(e^-) - \mathbf{x}(e^+) = \mathrm{C}(e^+) = -\mathrm{C}(e^-)\,. \tag{5.2}$$

In order to remove noise in the contour image, a 3×3 median filter is applied to C and the resulting edges are re-matched to form new contours. Choppy contours resulting from median filtering are repaired by vertically extending and connecting contour fragments of similar disparity. Fig. 5–2(e) shows the final contour image C_*.

5.2 Multihistogram segmentation

The segmentation scheme is based on two types of histograms: a histogram of disparities, and histograms of x coordinates, also known as the *horizontal signatures* of an image. The process is illustrated in Fig. 5–3.

First, from contour image C_* in Fig. 5–3(a), a disparity histogram is calculated by counting the contour segments at each differential disparity value, $\mathbf{d}(c)$, as shown in Fig. 5–3(b), top left. After 1D Gaussian smoothing, disparity ranges containing a single peak are extracted, ending at histogram valleys or zeroes. For example, two ranges of interest can be extracted from Fig. 5–3(a), one at $[1, 10]$ and the other $[11, 26]$.

Next, for each disparity range of interest, $\left[\mathbf{d}^i_{min}, \mathbf{d}^i_{max}\right]$, a horizontal signature of C_* is computed counting only line segments $\left\{c \mid \mathbf{d}^i_{min} \leq \mathbf{d}(c) \leq \mathbf{d}^i_{max}\right\}$. In Fig. 5–3(b), top right, the first histogram counts line segments of Fig. 5–3(a) with differential disparity in $[1, 10]$; the second, those in $[11, 26]$. Each horizontal signature is also smoothed and its peaks located. Small peaks, usually due to noise, are discarded.

Finally, adjacent peaks p^i_j and p^i_{j+1} in the i^{th} horizontal signature are grouped if

$$\left|\mathbf{x}(p^i_{j+1}) - \mathbf{x}(p^i_j)\right| < {\tau_{\mathrm{x}}}^i, \tag{5.3}$$

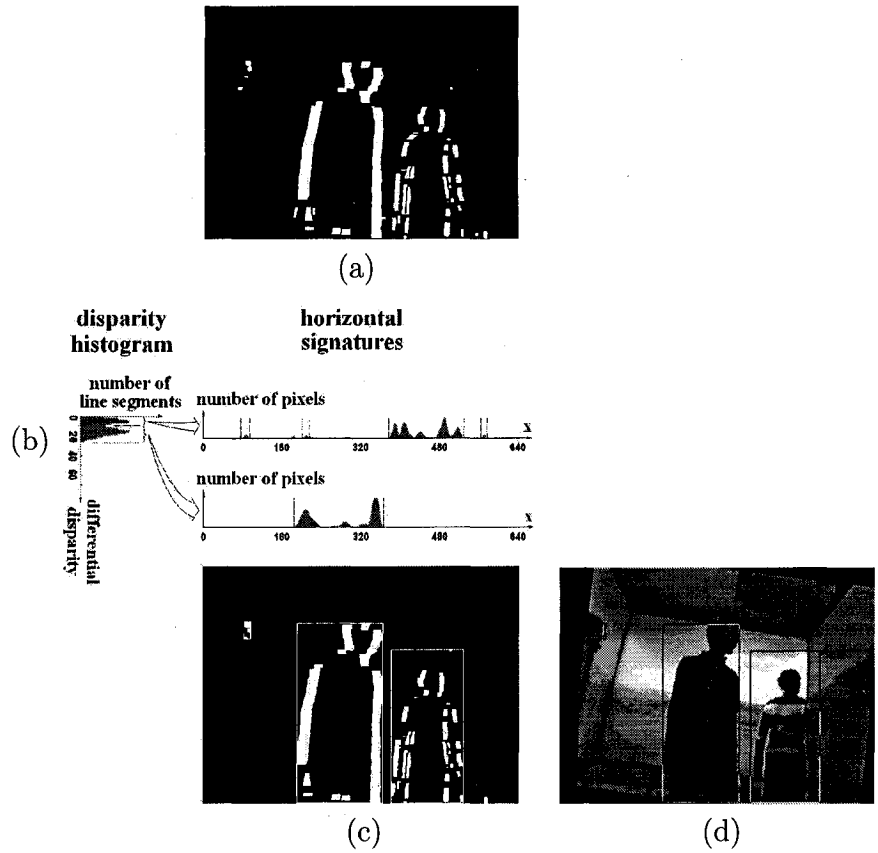


Figure 5–3: Multihistogram segmentation, left view. (a) Contour image in Fig. 5–2(f). (b) Left: disparity histogram with disparity ranges of interest indicated by dotted horizontal bars. Right: horizontal signatures of corresponding disparity ranges of interest. Peak grouping indicated by dotted vertical bars. (c) Object regions resulting from contour grouping, indicated by bounding boxes. (d) Objects identified in the scene.

where ${\tau_x}^i$ is a horizontal distance threshold. Grouped contours in each disparity range form object regions in the contour image, indicated by bounding boxes at the bottom of Fig. 5–3(b). Fig. 5–3(c) shows the results overlaid on the original image.

As the width of a foreground object is inversely proportional to the object-to-camera distance, which is in turn inversely proportional to the object disparity, the object width is linearly proportional to its disparity. This motivates the determination of the distance threshold ${\tau_x}^i$ for contour grouping by its corresponding disparity range, $[\mathbf{d}^i_{min}, \mathbf{d}^i_{max}]$:

$$ {\tau_x}^i = \kappa(\mathbf{d}^i_{min} + \mathbf{d}^i_{max})/2 + \varsigma, \tag{5.4} $$

where $\kappa = 3$ and $\varsigma = 20$ are tunable empirical parameters.

5.3 Experimental results

The disparity multihistogram segmentation method was tested in the *Shared Reality Environment.* Two cameras with fixed focal length were placed at a height of 1m, facing the screens. Grayscale image sequences were captured at a resolution of 640 × 480 pixels.

The method was validated first on an image sequence with static projected background in order to exclude interference from camera synchronization error (Fig. 5–4) and then on a sequence with moving video background having rapid changes in texture and illumination (Fig. 5–5). The quantitative analysis of the segmentation results is shown in Table 7–1 and Fig. 7–10 for the static background sequence and in Table 7–2 and Fig. 7–11 for the video background sequence. As can be seen, the proposed method is able to extract multiple foreground objects from a complex scene despite changing background lighting and texture. The rate of 'correct' segmentation, encompassing exact bounding box, noise enlarged bounding box, and partial object bounding box, is around 48% and 69% respectively on the two sequences.

Although the results are encouraging, problems are also evident. A summary of the problems with corresponding example frames appears in Table 5–1. First, the multihistogram approach to contour grouping is mechanical, without understanding the concept of objects. When an object falls in more than one disparity range in the disparity histogram, multiple bounding boxes containing partial objects result, as in frames 330 and 340 of Fig. 5–4 and frames 500 and 655 of Fig. 5–5. This is why the partial object category is dominant, constituting about 2/3 of the 'correct' segmentation, as shown in Fig. 7–10 and Fig. 7–11. Moreover, the resolution of object segmentation depends on the resolution of disparity range extraction from the disparity histogram and the peak distribution in the horizontal signatures of a contour image, which causes systematic difficulty in separating objects that are close in distance and depth, as in frames 120∼150 of Fig. 5–4 and frames 850 and 870 of Fig. 5–5. When the disparity difference between two objects does not result in distinct peaks in the disparity histogram, false grouping occurs even among objects not close in depth, as seen in frames 290 and 300 of Fig. 5–4 and frames

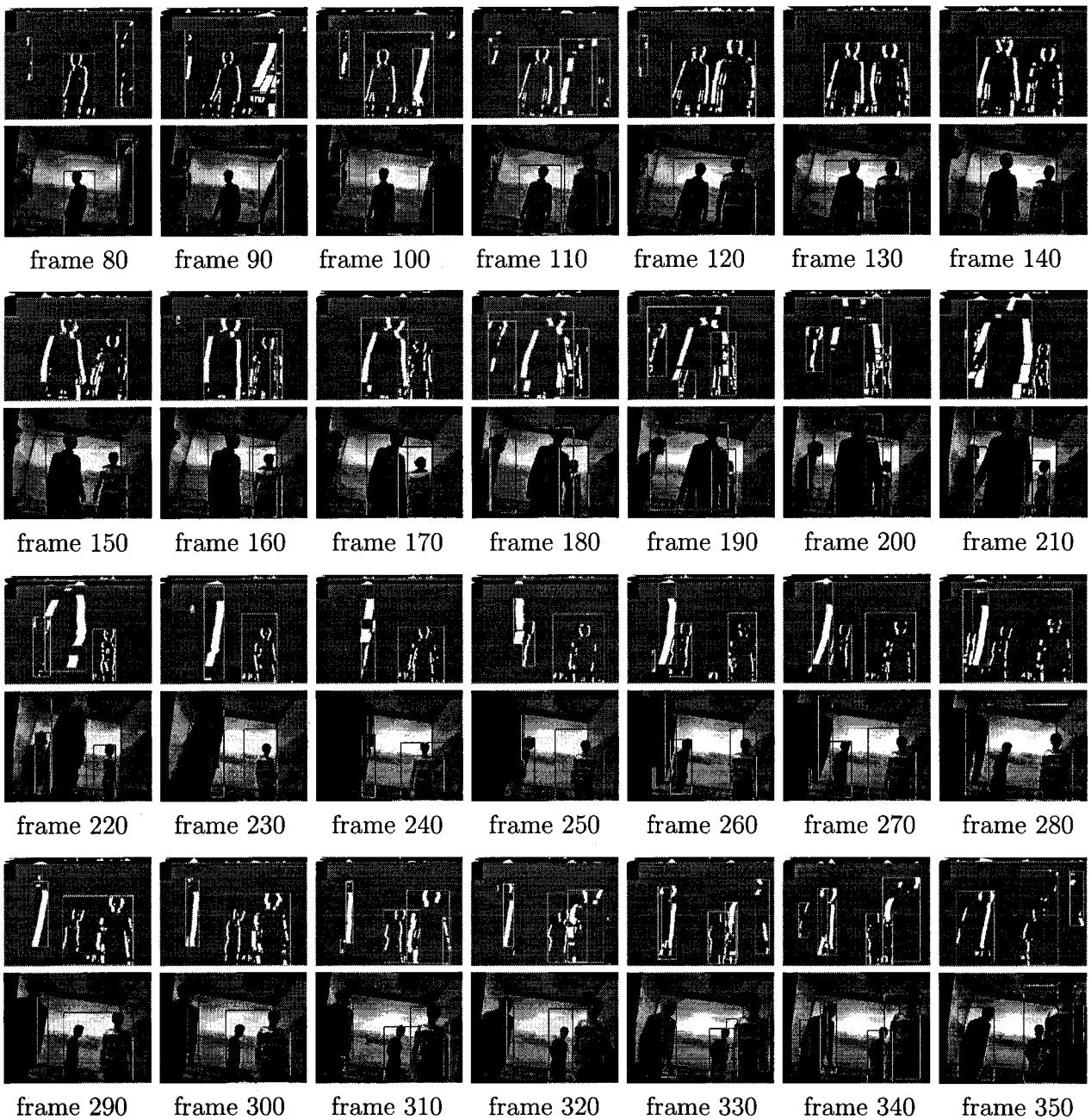


Figure 5–4: Disparity multihistogram segmentation results on a sequence with static background and three subjects.

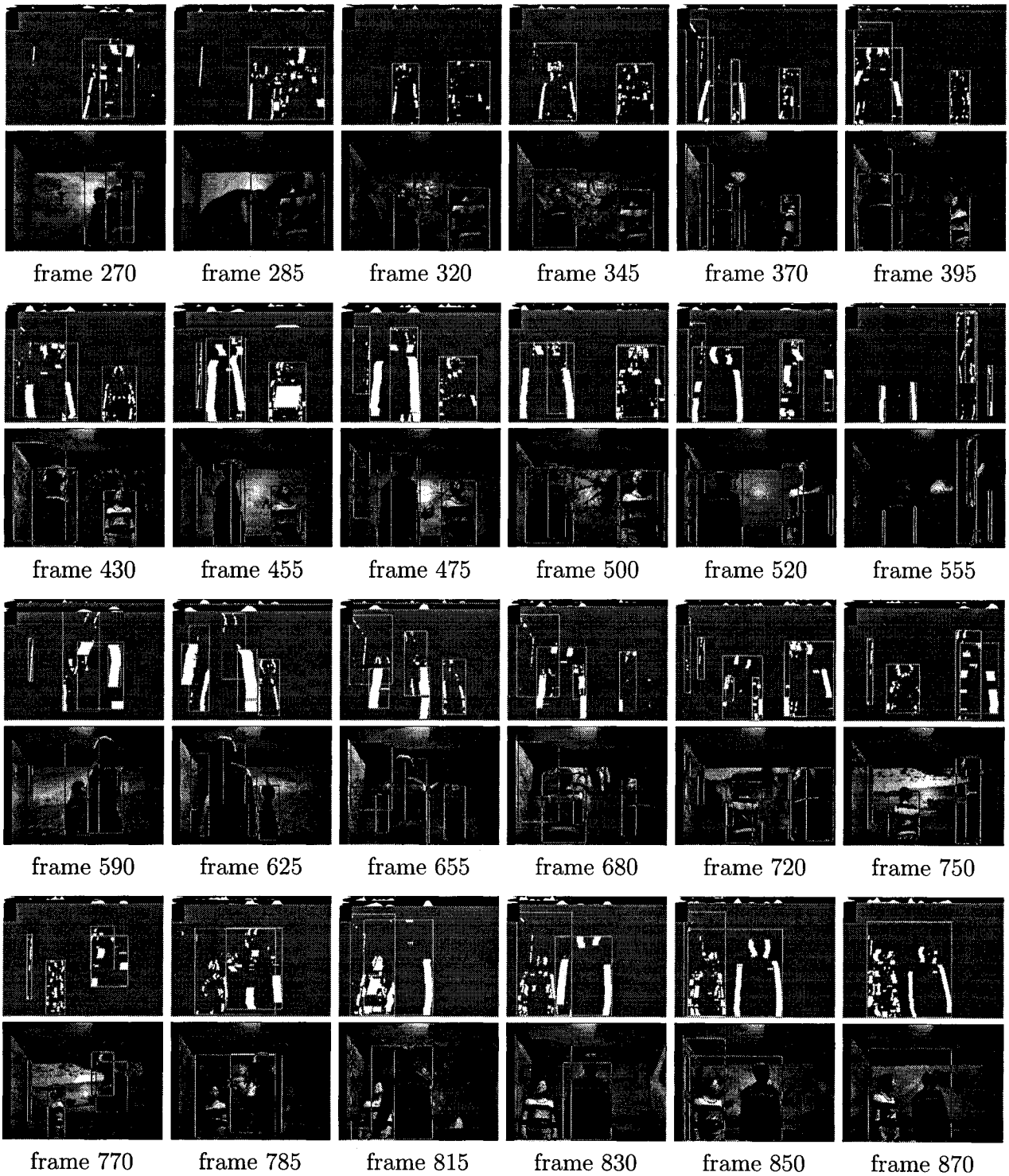


Figure 5–5: Disparity multihistogram segmentation results on a sequence with moving video background and two subjects.

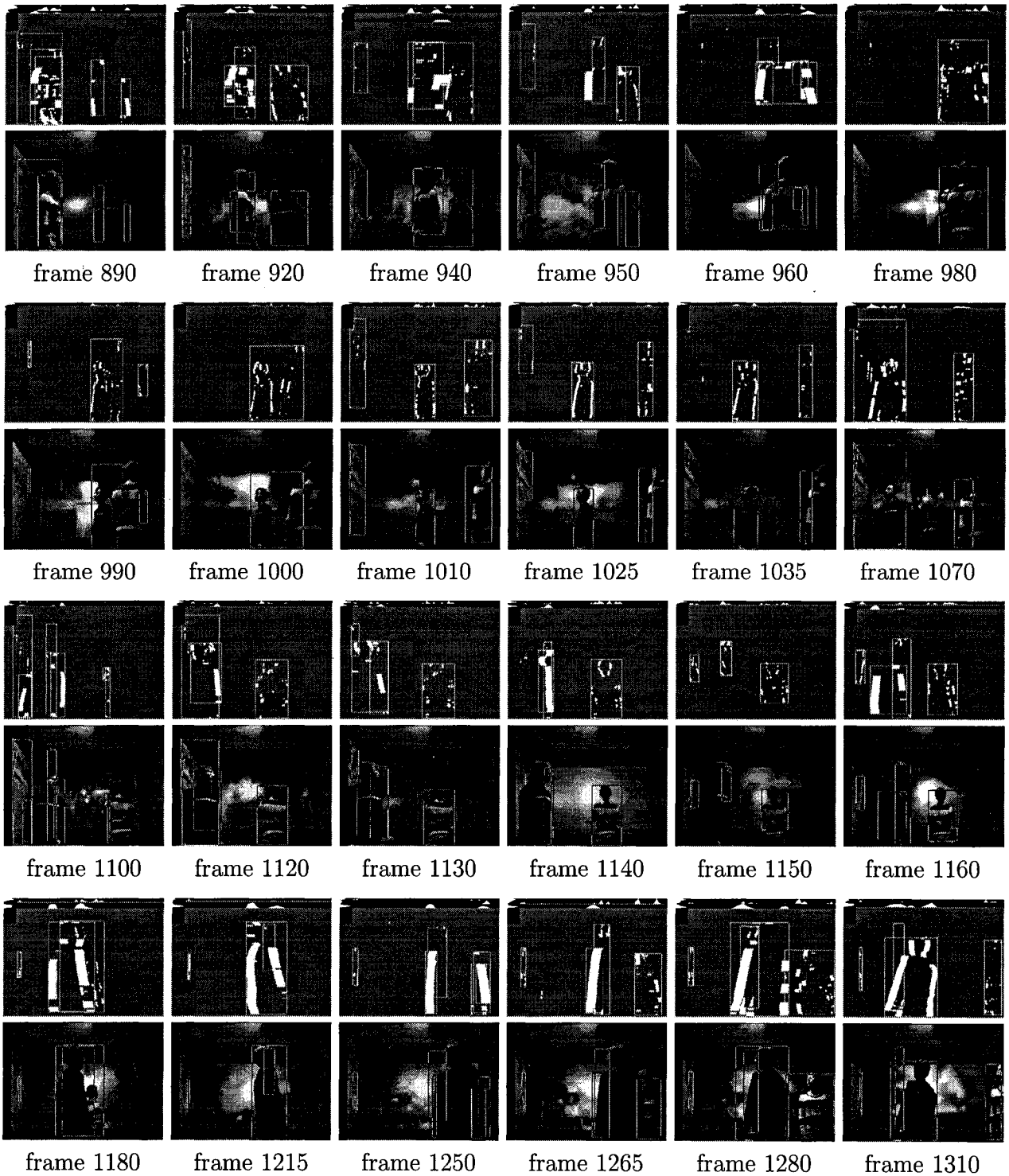


Figure 5–5: Disparity multihistogram segmentation results on a sequence with moving video background and two subjects. (cont.)

Table 5–1: Result analysis of disparity multihistogram segmentation.

remaining problems	example frames	
	static background Fig. 5–4	video background Fig. 5–5
multiple bounding boxes containing partial objects	330, 340	500, 655
difficulty in separating objects close in distance and depth	120~150	850, 870
disparity difference between objects does not cause peaks in disparity histogram	290, 300	285, 1000
object coalescing due to background noise and partial occlusion	90, 280	
false edge pairing		455, 940
object detection or grouping failure due to contour loss	130~170	555
false grouping due to partial occlusion	350	990, 1265
object partially out of view	240~320	1010~1035

285 and 1000 of Fig. 5–5. This explains the high proportion of 'object coalescing', about 42% and 21% of the total, in the result. In addition, remaining noise from background and internal object texture, and object self-occlusion, often yield false disparity information and confuse the peak distribution of the horizontal signatures. This may enlarge object bounding boxes, create false objects, or result in false grouping across multiple objects. These problems are obvious in frames 90 and 280 of Fig. 5–4.

Second, the quality of object segmentation relies on the qualities of contour extraction and contour grouping, which are determined by parameter tuning. The contour extraction result is sensitive to the choice of the edge threshold τ_{E}. The higher the threshold, the more unwanted structure is removed but the more useful contours are lost. Contour grouping is controlled by κ and ς, which are difficult to choose in a compromise between grouping partial objects and preventing object coalescing. Besides, empirical parameter tuning limits the system's adaptability to unknown scenes and is unsuitable for real-life applications.

Third, the median filter, although not completely successful in eliminating false edge pairing, as exemplified by frames 455 and 940 of Fig. 5–5, causes a loss of object boundary contours. This information loss either results in occasional failure to detect an object due to insufficient contour

evidence, as shown in frames 130~170 of Fig. 5–4 where the person on the left is not detected, or prevents successful contour grouping as in frame 555 of Fig. 5–5.

Fourth, even though the proposed method is somewhat able to handle partial occlusion between objects in cases such as in frames 200~270 of Fig. 5–4 and frames 590 and 815 of Fig. 5–5, this should not be seen as a reliable feature. The irregular shape of contour stripes within an occlusion area has often confused the segmenter, as in frame 350 of Fig. 5–4 and frames 990 and 1265 of Fig. 5–5, resulting in an overall worse performance (around 40% and 24% correct segmentation) compared to the no occlusion cases (51% and 77%).

Last, when an object moves partially out of view, the contours no longer provide an accurate object location, as shown in frames 240~320 of Fig. 5–4 and frames 1010~1035 of Fig. 5–5.

These problems motivate the investigation of approaches more focused on objects than the statistical properties of a scene.

CHAPTER 6
Contour Grouping Segmentation

The naive noise removal and contour grouping strategy of the disparity multihistogram approach resulted in systematically inaccurate segmentation results. This chapter presents a much improved disparity contour based scene segmenter, illustrated in Fig. 6–1. It employs a multi-evidential dual-threshold contour filter, statistical noise removal, and heuristic based contour grouping to overcome the weaknesses of the previous method and reduce its dependence on empirical parameter tuning.

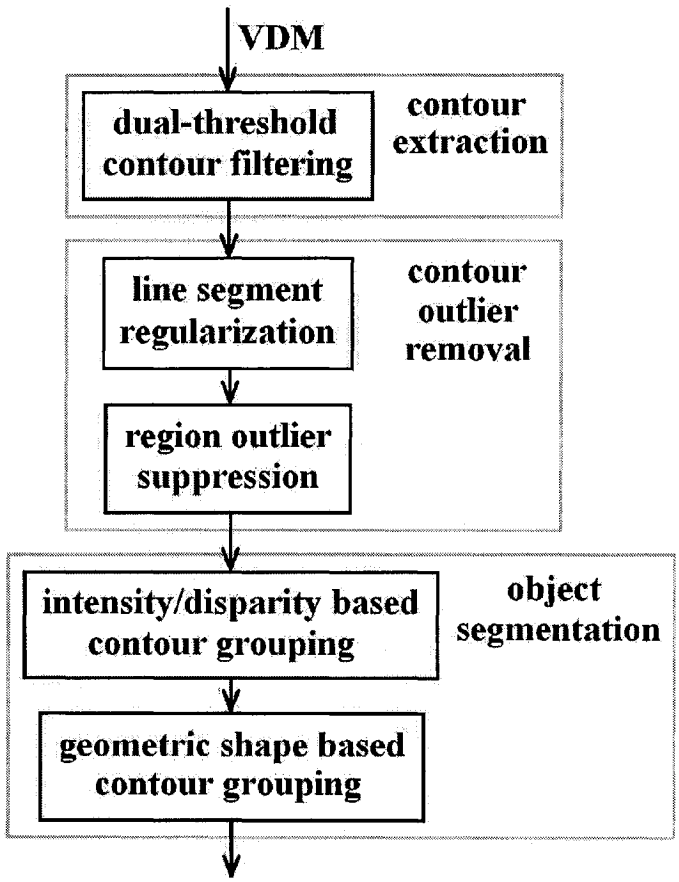


Figure 6–1: Contour grouping object segmentation scheme.

6.1 Contour extraction

In order to extract object boundary contours while suppressing unwanted structures caused by excessive foreground object internal texture and camera calibration or synchronization error, a dual-threshold coupled edge/intensity contour filter is developed.

6.1.1 Coupled edge/intensity contour extractor

Let D represent the difference image for either left or right view, as obtained from Eq. (4.3), an example of which is shown in Fig. 6–2(b), and E the edge image generated from D by the $[-1\ 1]$ edge operator. Positive and negative edge points in E that correspond to the left and right sides of a contour ridge in D, e^+ to the left of e^- on the same scanline, are paired if they satisfy the following three conditions:

$$\begin{aligned} &1.\ \ \mathrm{E}(e^+)\,/\,\max(\mathrm{E}) > \tau_{\mathrm{E}} \wedge \mathrm{E}(e^-)\,/\,\min(\mathrm{E}) > \tau_{\mathrm{E}};\\ &2.\ \ |\mathrm{E}(e^+) + \mathrm{E}(e^-)| < \varepsilon_{\mathrm{E}};\\ &3.\ \ \forall_{p\in[e^+,e^-]}\mathrm{D}(p) \geq \min(\mathrm{D}(e^+)\,,\mathrm{D}(e^-)). \end{aligned} \tag{6.1}$$

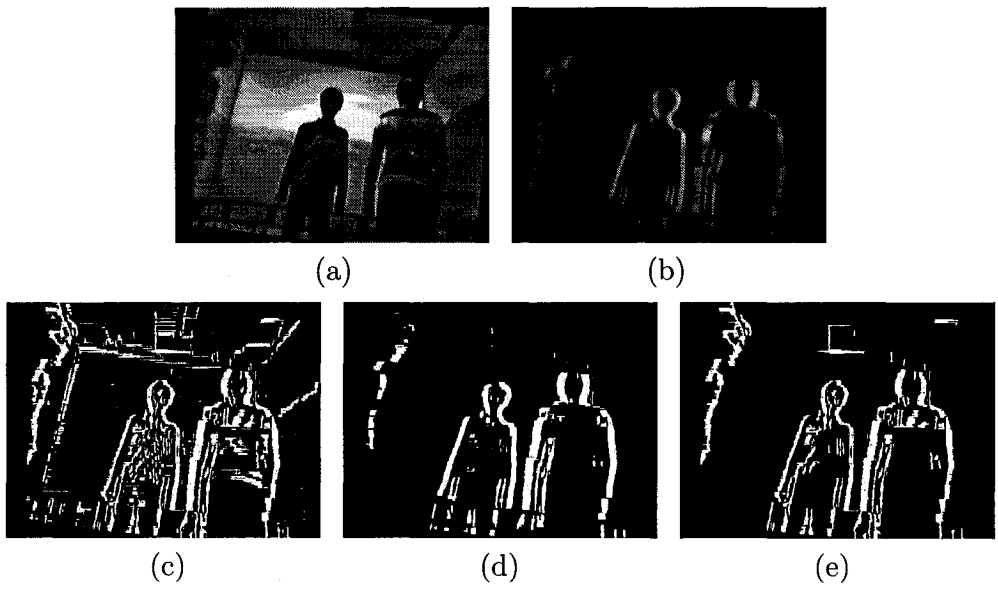


Figure 6–2: Contour extraction, left view. Contours displayed as binary line segments joining edge pairs. (a) Original image after image distortion removal and rectification. (b) Projection D of computed view difference.(c) Contour $\mathrm{C}_{\sqcup}$ extracted using low threshold $\tau_{\mathrm{E}} = \tau_{\sqcup}$. (d) Contour $\mathrm{C}_{\sqcap}$ extracted using high threshold $\tau_{\sqcap}$. (e) Merger C_* of (c) and (d) preserving useful object boundary contours and removing unwanted structure.

Compared to Eq. (5.1), the added third condition guarantees that the line segment $[e^+, e^-]$ corresponds to a high intensity contour ridge in D. Overlapping segments are resolved in favor of higher edge intensity.

The line segments, c, between the paired positive and negative edge points are the desired disparity contours. As in Eq. (5.2), this information is extracted and stored in a contour image, C, illustrated in Fig. 6–2(c).

6.1.2 Dual-threshold contour filter

As analyzed in Sect. 5.3, the above single contour extractor is sensitive to the edge threshold, τ_{E}. The higher the threshold, the more interesting contours are lost, but the less unwanted structure remains. Fig. 6–2(c) and (d) demonstrate contour images, $\mathrm{C}_{\sqcup}$ and $\mathrm{C}_{\sqcap}$, resulting from the use of a low, $\tau_{\sqcup}$, and a high, $\tau_{\sqcap}$, threshold, respectively. In order to alleviate this sensitivity, a dual-threshold contour filter is used and the advantages of the two different threshold values are combined.

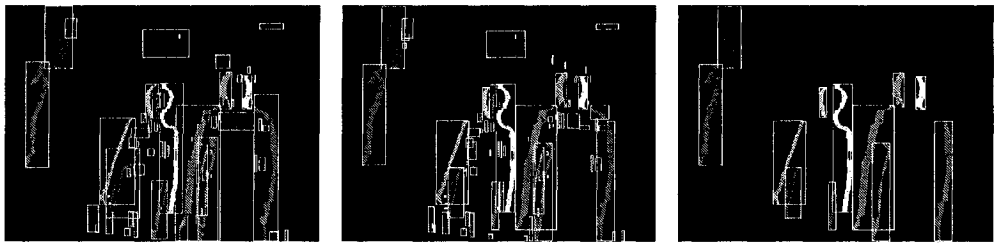
Connected contour line segments are grouped to form contour regions, R. Let $\mathrm{S}_{\sqcup}$ and $\mathrm{S}_{\sqcap}$ be the set of such contour regions in $\mathrm{C}_{\sqcup}$ and $\mathrm{C}_{\sqcap}$, respectively. A new contour image, C_*, is generated, as shown in Fig. 6–2(e), by preserving only $\mathrm{S}_{\sqcup}$ regions overlapping $\mathrm{S}_{\sqcap}$:

$$\mathrm{C}_* = \{\mathrm{R}_{\sqcup} \in \mathrm{S}_{\sqcup} \mid \exists_{\mathrm{R}_{\sqcap} \in \mathrm{S}_{\sqcap}} \mathrm{R}_{\sqcup} \cap \mathrm{R}_{\sqcap} \neq \emptyset\}\,. \tag{6.2}$$

In the implementation, $\tau_{\sqcup} = 0.015$ and $\tau_{\sqcap} = 0.05$. However, because of the robustness provided by the dual-threshold filtering, small variations to the threshold values have little impact on the results, and no further parameter tuning was required during testing.

6.2 Contour outlier removal

In order to remove the remaining noise in C_*, the global statistics of contour regions are calculated, and a statistical outlier removal method is employed. This process consists of two steps: regularization of line segments and suppression of contour region outliers.



(a) (b) (c)

Figure 6–3: Contour outlier removal, left view. Bounding boxes indicating contour regions, and brightness representing $\bar{\mathrm{I}}(\mathrm{R})$, the average region intensity. (a) Fig. 6–2(e) overlaid with bounding boxes. (b) Line segment outliers regularized. (c) Contour region outliers suppressed.

6.2.1 Line segment regularization

The first step eliminates line segment outliers within a contour region. Let R be a contour region containing $|\mathrm{R}|$ line segments, $c \in \mathrm{R}$ be a line segment consisting of $|c|$ pixels, and $p \in c$ be a pixel on c. The average intensity $\bar{\mathrm{I}}(c)$ of c can be expressed as

$$\bar{\mathrm{I}}(c) = \frac{\sum_{p \in c} \mathrm{D}(p)}{|c|}. \tag{6.3}$$

Assume that the average intensity $\bar{\mathrm{I}}(c)$ and differential disparity $\mathbf{d}(c)$ of c are independent and their distributions within R can be represented by a bivariate Gaussian:

$$\eta\left(\bar{\mathrm{I}}(c), \mathbf{d}(c); \bar{\mathrm{I}}(\mathrm{R}), \tilde{\mathbf{d}}(\mathrm{R}); \boldsymbol{\sigma}_{\mathrm{i}}(\mathrm{R}), \boldsymbol{\sigma}_{\mathrm{d}}(\mathrm{R})\right) \propto \frac{1}{\boldsymbol{\sigma}_{\mathrm{i}}(\mathrm{R})\boldsymbol{\sigma}_{\mathrm{d}}(\mathrm{R})} \exp\left(-\frac{(\bar{\mathrm{I}}(c) - \bar{\mathrm{I}}(\mathrm{R}))^2}{2\boldsymbol{\sigma}_{\mathrm{i}}(\mathrm{R})^2} - \frac{(\mathbf{d}(c) - \tilde{\mathbf{d}}(\mathrm{R}))^2}{2\boldsymbol{\sigma}_{\mathrm{d}}(\mathrm{R})^2}\right). \tag{6.4}$$

The mean values $\bar{\mathrm{I}}(\mathrm{R})$ and $\tilde{\mathbf{d}}(\mathrm{R})$ are

$$\bar{\mathrm{I}}(\mathrm{R}) = \frac{\sum_{c \in \mathrm{R}} \bar{\mathrm{I}}(c)}{|\mathrm{R}|} \quad \text{and} \quad \tilde{\mathbf{d}}(\mathrm{R}) = \frac{\sum_{c \in \mathrm{R}} \bar{\mathrm{I}}(c)\,\mathbf{d}(c)}{\sum_{c \in \mathrm{R}} \bar{\mathrm{I}}(c)}, \tag{6.5}$$

where $\tilde{\mathbf{d}}(\mathrm{R})$ is defined to be a weighted average disparity in order to give more weight to line segments with high $\bar{\mathrm{I}}(c)$. Fig. 6–3(a) shows contour regions in their average intensity. The deviations $\boldsymbol{\sigma}_{\mathrm{i}}(\mathrm{R})$ and $\boldsymbol{\sigma}_{\mathrm{d}}(\mathrm{R})$ are

$$\boldsymbol{\sigma}_{\mathrm{i}}(\mathrm{R}) = \sqrt{\frac{\sum_{c \in \mathrm{R}} (\bar{\mathrm{I}}(c) - \bar{\mathrm{I}}(\mathrm{R}))^2}{|\mathrm{R}|}} \quad \text{and} \quad \boldsymbol{\sigma}_{\mathrm{d}}(\mathrm{R}) = \sqrt{\frac{\sum_{c \in \mathrm{R}} \left(\mathbf{d}(c) - \tilde{\mathbf{d}}(\mathrm{R})\right)^2}{|\mathrm{R}|}}. \tag{6.6}$$

We can define an integrated intensity/disparity measure

$$\boldsymbol{\mu}_{\mathrm{I}}(c, \mathrm{R}) = |\bar{\mathrm{I}}(c) - \bar{\mathrm{I}}(\mathrm{R})|\,|\mathbf{d}(c) - \tilde{\mathbf{d}}(\mathrm{R})|, \tag{6.7}$$

and a coupled deviation

$$\tilde{\boldsymbol{\sigma}}(\mathrm{R}) = \boldsymbol{\sigma}_{\mathrm{i}}(\mathrm{R})\boldsymbol{\sigma}_{\mathrm{d}}(\mathrm{R}) = \frac{\sqrt{\sum_{c\in\mathrm{R}} (\bar{\mathrm{I}}(c) - \bar{\mathrm{I}}(\mathrm{R}))^2 \sum_{c\in\mathrm{R}} \left(\mathbf{d}(c) - \tilde{\mathbf{d}}(\mathrm{R})\right)^2}}{|\mathrm{R}|}. \tag{6.8}$$

Combining the following two conditions

$$\begin{aligned} &1.\ \ |\bar{\mathrm{I}}(c) - \bar{\mathrm{I}}(\mathrm{R})| > \sqrt{2}\boldsymbol{\sigma}_{\mathrm{i}}(\mathrm{R});\ \text{and} \\ &2.\ \ |\mathbf{d}(c) - \tilde{\mathbf{d}}(\mathrm{R})| > \sqrt{2}\boldsymbol{\sigma}_{\mathrm{d}}(\mathrm{R}), \end{aligned} \tag{6.9}$$

a contour line segment c is considered an outlier and removed from its region R if

$$\boldsymbol{\mu}_{\mathrm{I}}(c, \mathrm{R}) > 2\tilde{\boldsymbol{\sigma}}(\mathrm{R}). \tag{6.10}$$

The product in Eq. (6.7) is used instead of the Euclidean norm in order to bias towards preserving line segments where either $\bar{\mathrm{I}}(c)$ or $\tilde{\mathbf{d}}(\mathrm{R})$ is close to the mean. This was found to yield better results in practice. The value $\sqrt{2}$ in the thresholds of Eq. (6.9) is an empirical choice, but once set, the adaptive thresholding of Eq. (6.10) proves insensitive to variations of lighting and background texture in experiments.

Finally, a contour region disconnected due to segment removal is reconnected by interpolating missing segments from their neighbors. Region statistics, $\bar{\mathrm{I}}(\mathrm{R})$, $\tilde{\mathbf{d}}(\mathrm{R})$, and $\tilde{\boldsymbol{\sigma}}(\mathrm{R})$, are then updated according to Eq. (6.3)-(6.8). The result, which has smoother contour stripes, is shown in Fig. 6–3(b).

6.2.2 Contour region outlier suppression

The second step removes region outliers globally. Let S represent the set of |S| contour regions resulting from the previous step. Based on the observation that unwanted regions are usually small and have low average intensity, as can be observed in Fig. 6–3(b), a summed

intensity $\hat{\mathrm{I}}(\mathrm{R})$ is defined combining both attributes:

$$\hat{\mathrm{I}}(\mathrm{R}) = \bar{\mathrm{I}}(\mathrm{R})\,|\mathrm{R}|\,. \tag{6.11}$$

Another Gaussian distribution is used to model the deviation of all the regions in S with respect to the region with the largest summed intensity, R^*:

$$\eta\left(\hat{\mathrm{I}}(\mathrm{R}), \hat{\mathrm{I}}(\mathrm{R}^*), \hat{\sigma}(\mathrm{S})\right) \propto \frac{1}{\hat{\sigma}(\mathrm{S})} \exp\left(-\frac{(\hat{\mathrm{I}}(\mathrm{R}) - \hat{\mathrm{I}}(\mathrm{R}^*))^2}{2\hat{\sigma}(\mathrm{S})^2}\right), \tag{6.12}$$

where the deviation

$$\hat{\sigma}(\mathrm{S}) = \sqrt{\frac{\sum_{\mathrm{R}\in\mathrm{S}} (\hat{\mathrm{I}}(\mathrm{R}) - \hat{\mathrm{I}}(\mathrm{R}^*))^2}{|\mathrm{S}|}}. \tag{6.13}$$

A contour region R is considered an outlier and removed from the set S by another adaptive thresholding:

$$|\hat{\mathrm{I}}(\mathrm{R}) - \hat{\mathrm{I}}(\mathrm{R}^*)| > \hat{\sigma}(\mathrm{S}). \tag{6.14}$$

Region statistics are again updated. The result is shown in Fig. 6–3(c).

6.3 Contour grouping

Contour grouping gathers contour fragments that belong to the same object. Computing closed bounding contours of objects from contour fragments relies on contour grouping, which has been studied for many decades in the area of perceptual organization [49, 50]. However, reliable contour grouping requires a large amount of computation and is unsuitable for real-time applications. This section presents a simple grouping technique based on computed global information. It is again divided into two steps: a local intensity/disparity based grouping and a global geometric-shape-based grouping.

6.3.1 Integrated intensity/disparity grouping

This step is based on heuristics inspired by the Gestalt Principles of similarity and proximity. It groups a contour region to its neighbors based on essentially the integrated intensity/disparity measure of Eq. (6.7):

$$\mu_{\mathrm{I}}(\mathrm{R}_i, \mathrm{R}_j) = |\bar{\mathrm{I}}(\mathrm{R}_i) - \bar{\mathrm{I}}(\mathrm{R}_j)|\,|\tilde{\mathrm{d}}(\mathrm{R}_i) - \tilde{\mathrm{d}}(\mathrm{R}_j)|. \tag{6.15}$$

Now let $\top(\mathrm{R})$ and $\perp(\mathrm{R})$ be the top and bottom end contour line segments of R. The distance, $\boldsymbol{\delta}^{\top}(\mathrm{R}_i, \mathrm{R}_j)$, from the top end of R_i to any region R_j is defined by:

$$\boldsymbol{\delta}^{\top}(\mathrm{R}_i, \mathrm{R}_j) = \min_{p \in \top(\mathrm{R}_i),\, q \in \mathrm{R}_j} \|p - q\|\,, \tag{6.16}$$

where p, q are pixels and $\|\cdot\|$ is the Euclidean norm.

The set of candidate regions for grouping with R_i at the top end, $\mathrm{S}_i^{\top}$, is then obtained by applying adaptive thresholding similar to Eq. (6.10):

$$\mathrm{S}_i^{\top} = \left\{ \mathrm{R}_{j \neq i} \,\middle|\, \boldsymbol{\mu}_{\mathrm{I}}(\mathrm{R}_i, \mathrm{R}_j) < 2\tilde{\boldsymbol{\sigma}}(\mathrm{R}_i) + 2\tilde{\boldsymbol{\sigma}}(\mathrm{R}_j) \wedge \boldsymbol{\delta}^{\top}(\mathrm{R}_i, \mathrm{R}_j) < 2\tilde{\mathrm{d}}(\mathrm{R}_i) + 2\tilde{\mathrm{d}}(\mathrm{R}_j) \right\}. \tag{6.17}$$

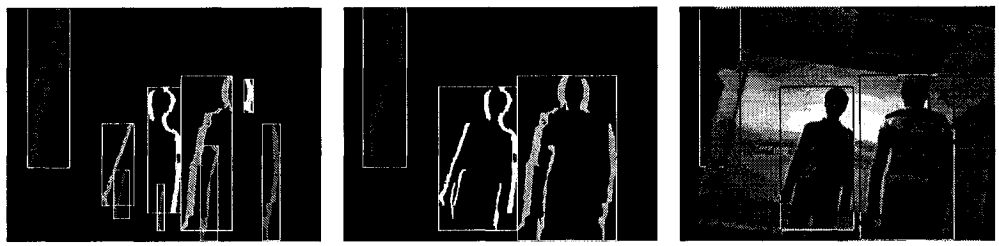
The first constraint guarantees that a candidate region is compatible with the current region in average intensity and disparity. The second adjusts the candidate search window according to the average disparities of two regions in question, as the width and height of an object is linearly proportional to its disparity.

Finally, if $\mathrm{S}_i^{\top}$ is nonempty, a region $\mathrm{R}_i^{\top}$ is chosen from $\mathrm{S}_i^{\top}$ to group with the top end of R_i. The choice of $\mathrm{R}_i^{\top}$ is made by selecting the region with a minimum $\boldsymbol{\mu}_{\mathrm{I}}$. For regions with equal $\boldsymbol{\mu}_{\mathrm{I}}$, the one with the smallest $\boldsymbol{\delta}^{\top}$ is chosen.

$$\mathrm{R}_i^{\top} = \arg \operatorname*{lex\,min}_{\mathrm{R} \in \mathrm{S}_i^{\top}} \left\langle \boldsymbol{\mu}_{\mathrm{I}}(\mathrm{R}_i, \mathrm{R}), \boldsymbol{\delta}^{\top}(\mathrm{R}_i, \mathrm{R}) \right\rangle, \tag{6.18}$$

where lex min means $\boldsymbol{\mu}_{\mathrm{I}}$ and $\boldsymbol{\delta}^{\top}$ are minimized in the given priority order. The same calculation is performed at the bottom end of R_i using the analogous distance function $\boldsymbol{\delta}^{\perp}(\mathrm{R}_i, \mathrm{R}_j)$ to locate and merge a bottom end candidate, $\mathrm{R}_i^{\perp}$.

Contour region labeling takes place in a single step, after the best merge candidates have been located for each region. Fig. 6–4(a) shows the final result, with bounding boxes indicating contour groups.



(a) (b) (c)

Figure 6–4: Contour grouping, left view. (a) Neighboring contour regions grouped based on integrated intensity/disparity measure μ_{I}. (b) Contour regions grouped based on convexity, μ_{C}. (c) Objects as located by grouped contours.

6.3.2 Geometric shape based grouping

Based on the heuristic that most real-life objects have a tendancy to global convexity due to shape or perspective, a simple convexity measure is defined for grouping contour regions with compatible convex-outwards shapes, which is consistent with the Gestalt Principle of closure.

As described in Section 6.1, a contour piece is produced by connecting line segments between pairs of positive and negative edge points. As in Fig. 6–5, let $x = \mathbf{f}^{+}(y)$ be the line joining the left extrema of the end segments of a contour region R, and $x = \mathbf{f}^{-}(y)$ that joining the right extrema, both represented as functions of y-coordinate. The convexity measure of the region, $\mu_{\mathrm{C}}(\mathrm{R})$, is defined as:

$$\mu_{\mathrm{C}}(\mathrm{R}) = \sum_{c \in \mathrm{R}} \operatorname{sgn}\left(\mathbf{x}(c^{+}) - \mathbf{f}^{+}(\mathbf{y}(c^{+}))\right) + \operatorname{sgn}\left(\mathbf{x}(c^{-}) - \mathbf{f}^{-}(\mathbf{y}(c^{-}))\right), \tag{6.19}$$

where c^{+} and c^{-} are the left and right endpoints of the contour line segment c, respectively. It can easily be shown that $\mu_{\mathrm{C}}(\mathrm{R}) < 0$ if R is overall convex towards the left, suggesting that it is

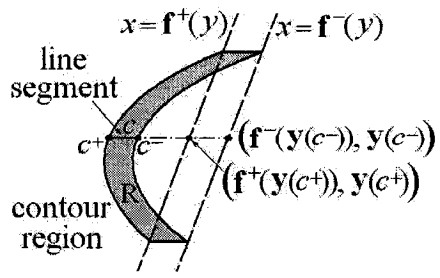


Figure 6–5: Computing the convexity of a contour stripe.

the left side of an object, and $\mu_C(\mathrm{R}) > 0$ if R is overall convex towards the right, suggesting that it falls on the right. The algorithm then groups contours classified as left or right side to find object regions, with result shown in Fig. 6–4(b). Fig. 6–4(c) shows the objects identified in the scene.

6.4 Experimental results

The contour grouping segmentation approach was validated on the same two image sequences used in Sect. 5.3, with sample images shown in Fig. 6–6 and Fig. 6–7 and quantitative analysis illustrated in Table 7–1, Table 7–2, Fig. 7–10 and Fig. 7–11. None of the parameters required empirical tuning during testing to achieve the demonstrated performance.

Comparing these results against those of the simpler multihistogram approach, the proportion of 'correct' segmentation has increased, to approximately 70% on the static background sequence and 68% on the video background sequence, due to a number of improvements. First, by employing multi-evidential contour extraction and adaptive noise removal, the contour grouping approach is able to preserve more useful contour information at object boundaries while removing unwanted structures in the background. This reduces object detection failure rate from 6% in the static background case and 0.3% in the video background case to 2.6% and 0.2%, respectively. This result can be seen by comparing frames 130, 140 and 170 of Fig. 6–6 and frame 555 of Fig. 6–7 against the same frames in Fig. 5–4 and Fig. 5–5.

Second, the contour grouping approach makes better use of object coherence and yields more accurate object locations by providing tighter bounding boxes, such as in frames 110 and 320 of Fig. 6–6 and frames 625, 920, 940, 1000 and 1280 of Fig. 6–7. This significantly increases the proportion of 'accurate' segmentation from the 5% and 16% with the multihistogram method to 43% and 42%. The improvement is especially obvious when an object is partially out of view, as in frames 250 and 270 of Fig. 6–6 and frames 1010~1035 of Fig. 6–7, or when two objects are close in distance and depth, as in frames 120~150 of Fig. 6–6 and frames 850 and 870 of Fig. 6–7.

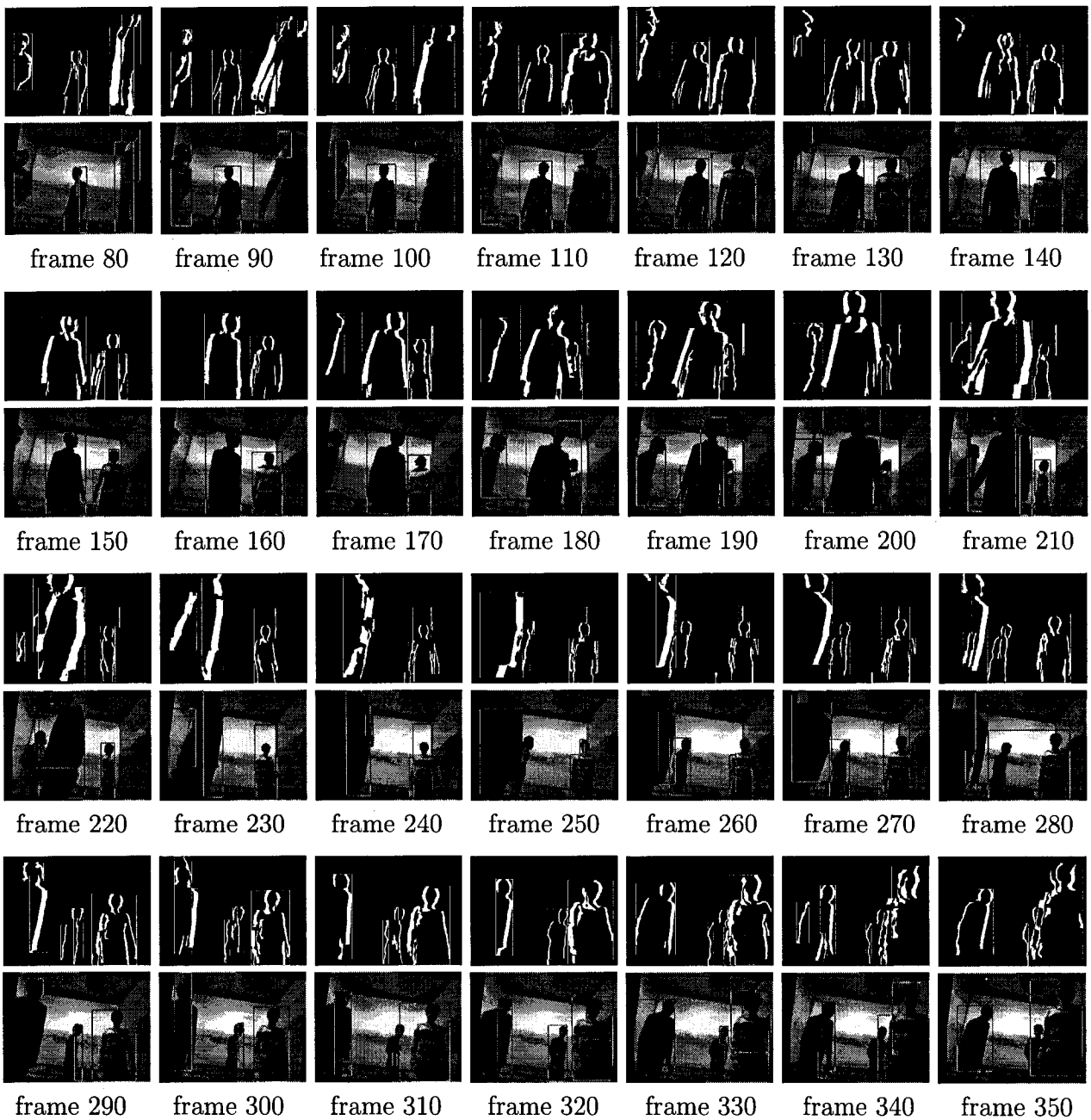


Figure 6–6: Contour grouping segmentation results on a sequence with static background and three subjects.

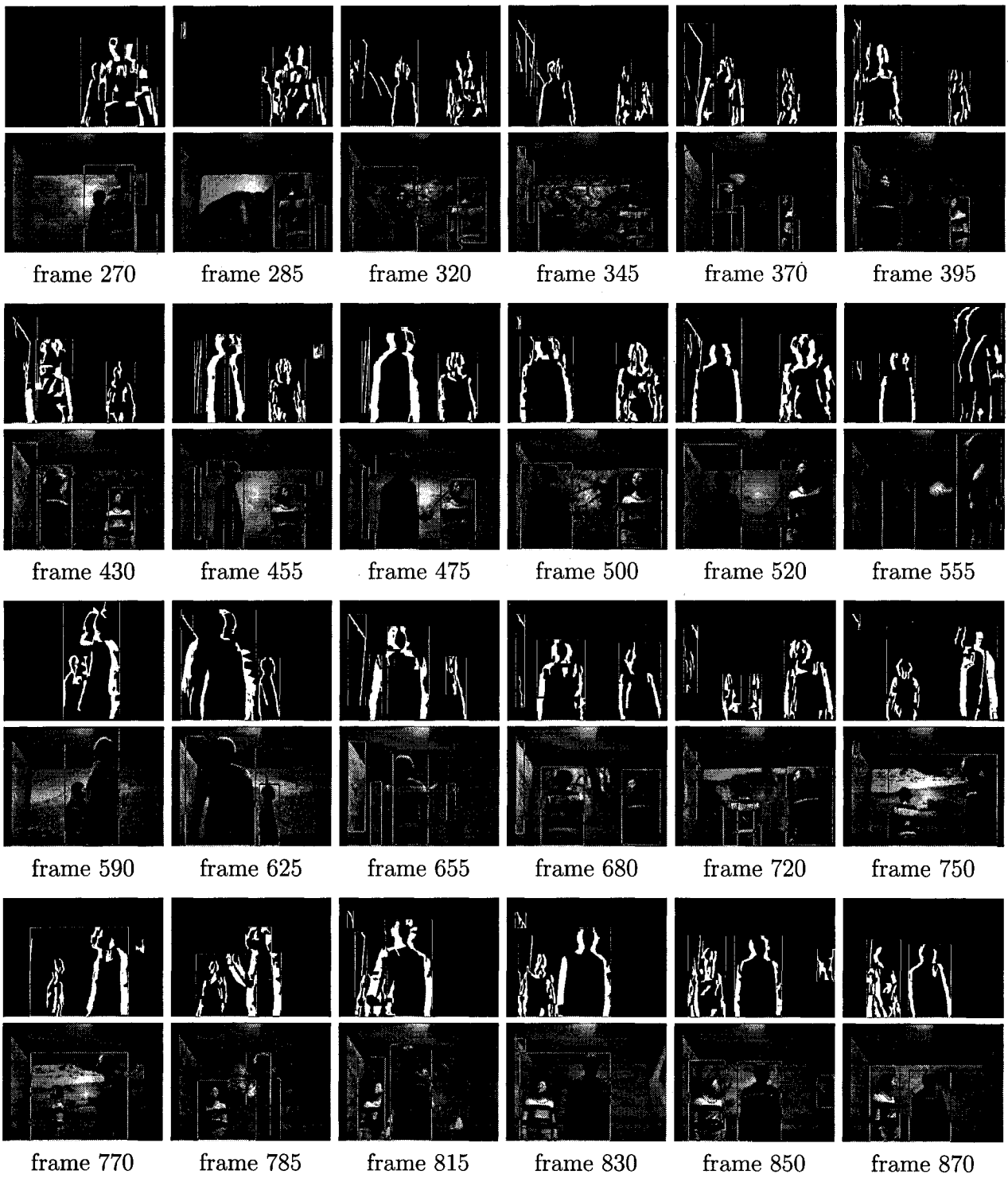


Figure 6–7: Contour grouping segmentation results on a sequence with moving video background and two subjects.

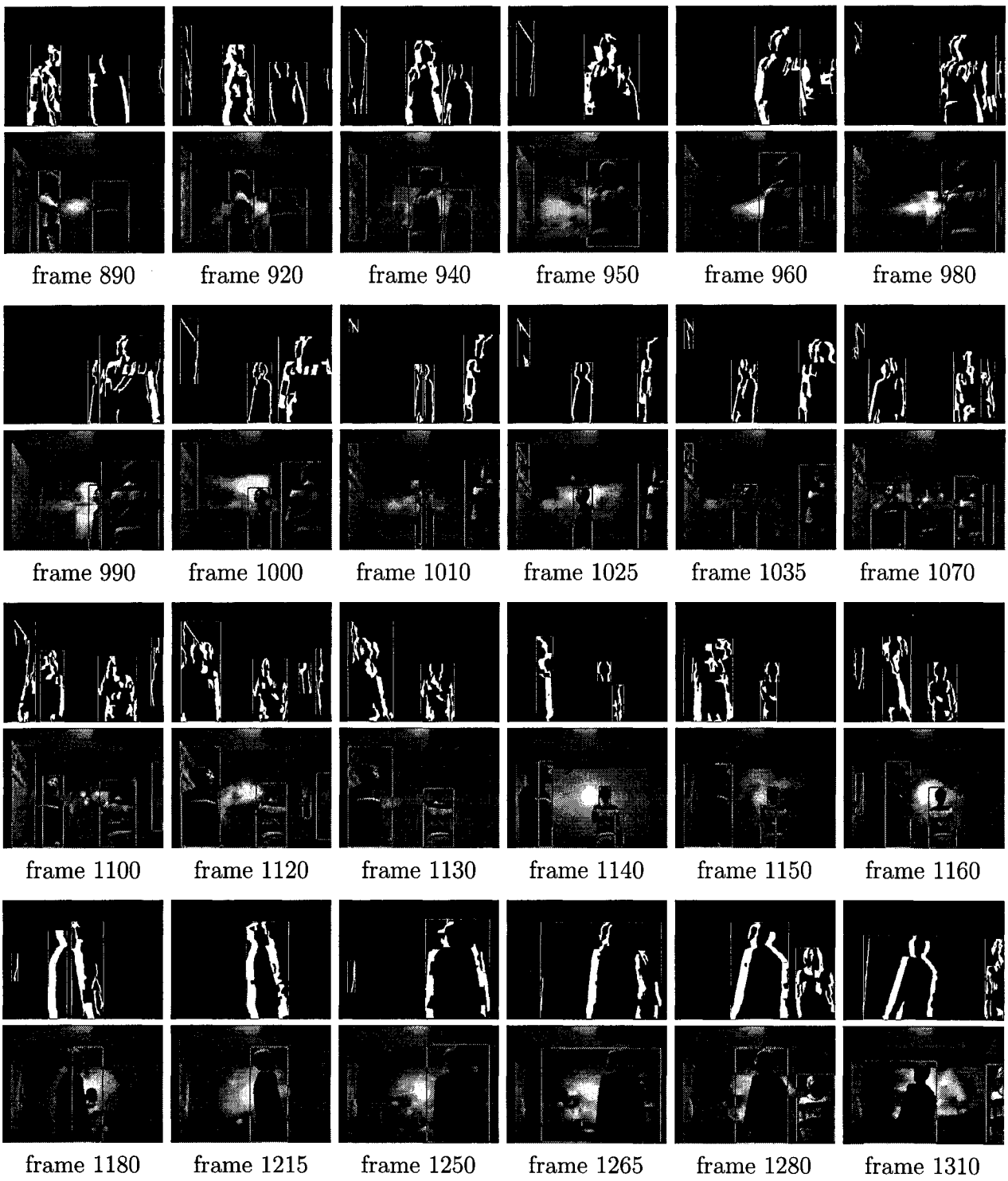


Figure 6–7: Contour grouping segmentation results on a sequence with moving video background and two subjects. (cont.)

Table 6–1: Result analysis of contour grouping segmentation.

improvements over disparity multihistogram approach	**example frames**	
	static background Fig. 5–4 vs. Fig. 6–6	**video background Fig. 5–5 vs. Fig. 6–7**
preserve contour information to reduce detection or grouping failure	130, 140, 170	555
use object coherence to yield more accurate object locations	110, 320	625, 920, 940,1000,1280
handle objects partially out of view	250, 270	1010∼1035
distinguish objects close in distance and depth	120∼150	850, 870
remaining problems	**example frames**	
	static background Fig. 6–6	**video background Fig. 6–7**
wrong convexity due to concavities	80, 290∼310	455, 785, 1010
wrong convexity due to imperfect contour extraction		655, 720
wrong convexity due to background noise, object texture or occlusion	330∼350	270∼370
coalescing due to failure to judge contour grouping completion	280	750, 770, 830
object falsely suppressed	150, 160	

While overall performance is good, the inherent shortcomings of contour grouping segmentation are obvious, mostly due to the convexity assumption on object shape. This assumption is not applicable to an object with a truly concave shape. Moreover, most objects in the real world contain local concavities, which, under certain circumstances, cause grouping errors, as for example in frames 80 and 290∼310 of Fig. 6–6 and frames 455, 785 and 1010 of Fig. 6–7. Even with convex objects, the convexity value, $\mu_C(R)$, of a contour stripe depends on the shape of the extracted contour and may exhibit the wrong sign due to imperfect contour extraction, resulting in a grouping failure, for example in frames 655 and 720 of Fig. 6–7. This problem is made worse by the remaining noise from the background and within an object, as well as partial occlusions, including self-occlusions. These can be observed in frames 330∼350 of Fig. 6–6 and frames 270∼370 of Fig. 6–7. Furthermore, the over-simplified global definition of the convexity measure is incapable of judging when the grouping of an object is complete, which is why false

grouping across multiple objects occurs in frame 280 of Fig. 6–6 and frames 750, 770 and 830 of Fig. 6–7. For the above reasons, 'object coalescing' remains the major cause (at around 70%) of 'incorrect' segmentation in this approach.

Another source of error is the statistical contour outlier removal. Although effective in removing unwanted structures, this process may falsely suppress small objects with low contour intensity, as with the face on the left in frames 150 and 160 of Fig. 6–6.

Table 6–1 summarizes the improvements and remaining problems of the contour grouping segmentation approach. Our final solution attempts to address these issues by further exploiting the properties of disparity contours.

CHAPTER 7
Disparity Verification Segmentation

The inherent weakness of the contour grouping segmentation scheme is its convexity assumption about object shape, which is not always true in a real environment. Additionally, the over-simplified convexity measure is not able to provide information as to whether all the contour fragments of an object have been collected, causing false grouping across different objects. In order to overcome this weakness, this chapter goes back to the idea of *background hypothesis falsification*, explores more features of *disparity contours*, and presents a disparity verification based contour grouping approach.

A diagram of the revised system is shown in Fig. 7–1. This new grouping process makes use of the results from both *background hypothesis falsification* and *foreground hypothesis verification*, and not only improves the segmentation results but also broadens the applicability of the algorithm. In preparation for explaining *foreground hypothesis verification*, this chapter starts by showing how to calculate the disparity of a foreground object.

7.1 Foreground disparity calculation

Foreground disparity can be calculated once disparity contours are extracted as described in Sect. 6.1. Let c be a contour line segment of length $|c|$ in the left view and c^+ and c^- its left and right end points. From Fig. 4–2 and Eq. (4.6), we have

$$\mathbf{d}(c) = |c| = \mathbf{x}_{\text{L}}(c^-) - \mathbf{x}_{\text{L}}(c^+) = \mathbf{d}_{\text{F}}(c^-) - \mathbf{d}_{\text{B}}(c^+), \tag{7.1}$$

where $\mathbf{d}(c)$ is the differential disparity between the background and foreground. We can rewrite this equation, while simplifying the notation without introducing ambiguity, as:

$$\mathbf{d}_{\text{F}}(c) = \mathbf{d}_{\text{B}}(c) + \mathbf{d}(c), \tag{7.2}$$

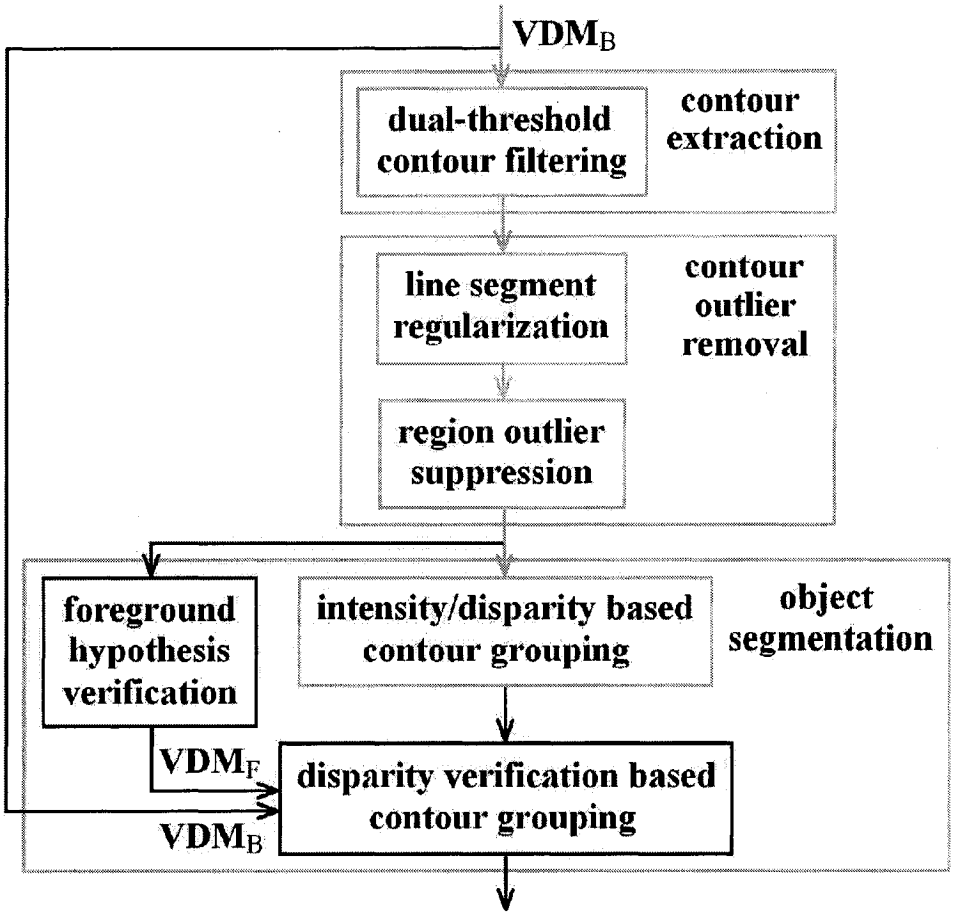


Figure 7–1: Disparity verification object segmentation scheme.

which calculates foreground disparity at the boundary point given background disparity and differential disparity. Let R be a contour region containing |R| such line segments. The average foreground disparity of R is obtained by:

$$\bar{\mathrm{d}}_{\mathrm{F}}(\mathrm{R}) = \frac{\sum_{c \in \mathrm{R}} \mathrm{d}_{\mathrm{F}}(c)}{|\mathrm{R}|}. \tag{7.3}$$

Similarly, if O is a foreground object containing |O| such contour regions, then the average foreground disparity of O is

$$\bar{\mathrm{d}}_{\mathrm{F}}(\mathrm{O}) = \frac{\sum_{\mathrm{R} \in \mathrm{O}} \bar{\mathrm{d}}_{\mathrm{F}}(\mathrm{R})}{|\mathrm{O}|}. \tag{7.4}$$

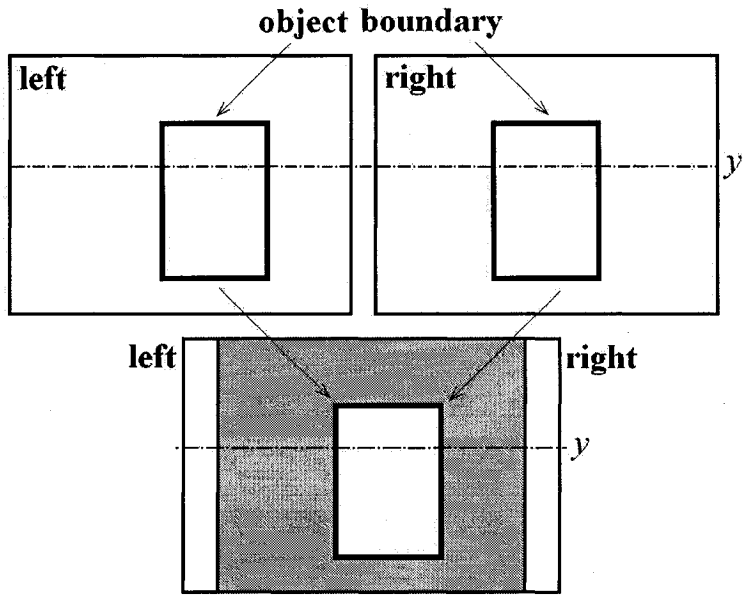


Figure 7–2: The ideal case of foreground hypothesis verification.

7.2 Foreground hypothesis verification (FHV)

The essence of *background hypothesis falsification* is disparity verification, which can be applied to foreground disparity as well. Assume the depth range of a foreground object O is much smaller than the object-to-camera distance and the object disparity can be approximated by $\bar{\mathbf{d}}_{\mathrm{F}}(\mathrm{O})$. Similarly to Sect. 4.1, if a pair of images is hypothesized to be of the foreground O alone, the block-matching operation on the two images

$$\mathrm{VDM}_{\mathrm{F}}(x_{\mathrm{L}}, x_{\mathrm{R}}, y) = \sum_{u,v} \left| \mathrm{V}_{\mathrm{L}}(x_{\mathrm{L}}+u, y+v) - \mathrm{V}_{\mathrm{R}}(x_{\mathrm{R}}+u, y+v) \right|, \text{ where } x_{\mathrm{L}} - x_{\mathrm{R}} = \bar{\mathbf{d}}_{\mathrm{F}}(\mathrm{O}), \quad (7.5)$$

ideally produces a high value in the background area and zero in the foreground object area, as illustrated in Fig. 7–2. In reality, the result depends again on the self-correlation of foreground texture, of background texture, the visual difference between the background and foreground, and between different foreground objects.

Although both *background hypothesis falsification* and *foreground hypothesis verification* are sensitive to texture self-correlation, the subtraction of the two results can generate more robust values. Assuming sufficient visual difference between background and foreground, Fig. 7–3

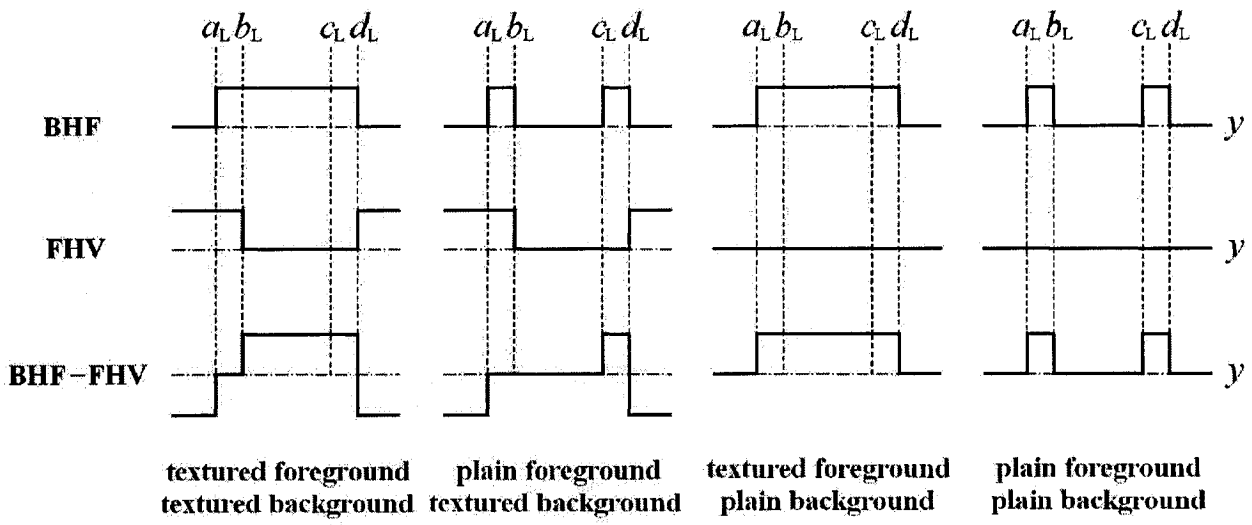


Figure 7–3: The y^{th} scanline of left projection $\mathrm{D_L}$ of VDM, where (a_L, y), (b_L, y), (c_L, y), (d_L, y) are the end points of disparity contour line segments, as in Fig. 4–3. Top: $\mathrm{D_L}$ from background hypothesis falsification (BHF); middle: $\mathrm{D_L}$ from foreground hypothesis verification (FHV); bottom: $\mathrm{D_L}$ from the subtraction of BHF and FHV.

demonstrates VDM results on a scanline corresponding to whether the background or foreground is textured or plain. As can be observed, the subtraction of *background hypothesis falsification* and *foreground hypothesis verification* results in overall higher values within the foreground area between the two contour stripes than in the background area. Due to the movement of foreground objects, it is statistically rare that the entire background and foreground regions remain textureless over time in a real environment. Hence, a reliable foreground confidence measure can be defined based on the subtraction results.

7.3 Foreground confidence and disparity contour direction

Let $\ltimes(\cdot)$ and $\rtimes(\cdot)$ be functions that yield the left and right neighbors of an element, respectively. Let the left neighbor $\ltimes(c)$ of a contour segment $c \in \mathrm{R}$ be the rightmost contour line segment to the left of c on the same scanline, as illustrated in Fig. 7–4. The left neighborhood of R can be constrained by left border points, $\vdash(c)$, defined to be either $\ltimes(c)^-$, the right end point of $\ltimes(c)$, or a distance threshold $\tau_\bowtie$ to the left of c^+, whichever is closest (in this case, rightmost):

$$\vdash(c) = \text{right}\left(\left(\mathbf{x}(c^+) - \tau_\bowtie, \mathbf{y}(c)\right),\ \ltimes(c)^- + 1\right), \tag{7.6}$$

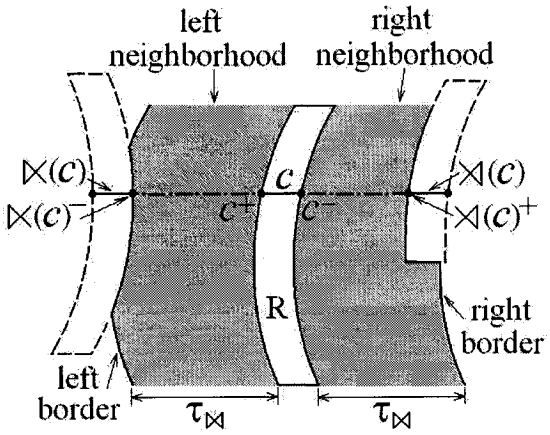


Figure 7–4: Neighborhood of contour region R for computing disparity direction.

where $\ltimes(c)^-$ is offset by 1 so that border points do not fall in a contour region. The *foreground confidence* $\mu_{\ltimes}(\mathrm{R})$ of R's left neighborhood can thus be defined by the subtraction of *background hypothesis falsification* (left hand of numerator) and *foreground hypothesis verification* (right hand of numerator), normalized by the left neighborhood area (denominator):

$$\mu_{\ltimes}(\mathrm{R}) = \frac{\sum_{c\in\mathrm{R}} \sum_{p\in[\vdash(c),c^+)} \mathrm{D_L}(\mathbf{x}(p),\mathbf{y}(p)) - \left|\mathrm{V_L}(\mathbf{x}(p),\mathbf{y}(p)) - \mathrm{V_R}\left(\mathbf{x}(p) - \bar{\mathbf{d}}_{\mathrm{F}}(\mathrm{R}),\mathbf{y}(p)\right)\right|}{\sum_{c\in\mathrm{R}} \mathbf{x}(c^+) - \mathbf{x}(\vdash(c))}, \quad (7.7)$$

where $\mathrm{D_L}$ is the left projection of $\mathrm{VDM_B}$ resulting from *background hypothesis falsification.* Similarly, the foreground confidence $\mu_{\rtimes}(\mathrm{R})$ of R's right neighborhood can be defined.

The *disparity contour direction* $\mu_{\mathrm{D}}(\mathrm{R})$, which determines the grouping direction of R when locating foreground objects, can be expressed by:

$$\mu_{\mathrm{D}}(\mathrm{R}) = \mu_{\ltimes}(\mathrm{R}) - \mu_{\rtimes}(\mathrm{R}). \quad (7.8)$$

According to Fig. 7–3, R is on the left boundary of an object if $\mu_{\mathrm{D}}(\mathrm{R}) < 0$, on the right boundary of an object if $\mu_{\mathrm{D}}(\mathrm{R}) > 0$, within an object or background or both the object and background are textureless if $\mu_{\mathrm{D}}(\mathrm{R}) = 0$. As noted in Section 7.2, the last case rarely persists, due to expected foreground movement.

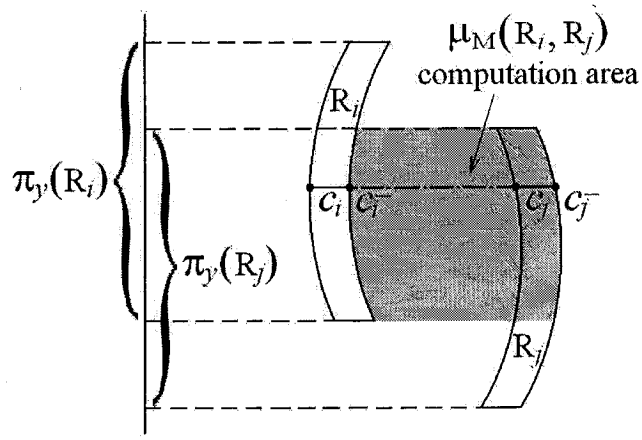


Figure 7–5: Geometric relations between contour regions R_i and R_j.

7.4 Disparity contour overlap, blocking, and match

In order to group contour regions that belong to the same object, three types of relationships between neighboring contour regions are analyzed. Given two contour regions R_i and R_j, as shown in Fig. 7–5, the overlap of R_j onto R_i is defined by an *overlap measure* $\mu_{\natural}(R_i, R_j)$:

$$\mu_{\natural}(R_i, R_j) = \frac{|\pi_y(R_i) \cap \pi_y(R_j)|}{|\pi_y(R_i)|}, \tag{7.9}$$

where $\pi_y(R)$ is the projection of R on the y-axis. Hence, we have

1. **Overlap**: R_i overlaps with R_j iff $\mu_{\natural}(R_i, R_j) > \tau_{\natural}$,
2. **Blocking**: R_i is blocked by R_j iff $\mu_{\natural}(R_i, R_j) > 1 - \tau_{\natural}$,

where $\tau_{\natural}$ is an overlap threshold. If R_i and R_j overlap with $\mu_D(R_i) < 0$ and $\mu_D(R_j) > 0$, a *matching cost* between R_i and R_j can be defined based on *foreground hypothesis verification*, normalized by the overlapping region between R_i and R_j:

$$\mu_M(R_i, R_j) = \frac{\sum_{c_i \in R_i, c_j \in R_j, y(c_i)=y(c_j)} \sum_{p \in [c_i^-, c_j^-]} \left| V_L(\mathbf{x}(p), \mathbf{y}(p)) - V_R\left(\mathbf{x}(p) - \bar{\mathbf{d}}_F(R_i, R_j), \mathbf{y}(p)\right) \right|}{\sum_{c_i \in R_i, c_j \in R_j, y(c_i)=y(c_j)} 1 + \mathbf{x}(c_i^-) - \mathbf{x}(c_j^-)}, \tag{7.10}$$

where

$$\bar{\mathbf{d}}_F(R_i, R_j) = \frac{|R_i|\, \bar{\mathbf{d}}_F(R_i) + |R_j|\, \bar{\mathbf{d}}_F(R_j)}{|R_i| + |R_j|}. \tag{7.11}$$

Therefore, the third type of relationship can be described as:

3. **Match**: R_i and R_j match iff $\boldsymbol{\mu}_{\mathrm{M}}(\mathrm{R}_i, \mathrm{R}_j) < \tau_{\mathrm{M}}$,

where τ_{M} is a matching cost threshold. An *object cost* $\boldsymbol{\mu}_{\mathrm{M}}(\mathrm{O})$ can also be defined in the same manner using the object's peripheral contours on the left and right sides.

7.5 Disparity verification based contour grouping

As seen in Fig. 6–1 and Fig. 7–1, disparity verification segmentation adopts the same contour extraction and outlier removal strategy as contour grouping segmentation. The difference lies in its global contour grouping step, which is based on disparity verification.

Let S_* represent the set of $|\mathrm{S}_*|$ contour regions after removing contour outliers, as in Sect. 6.2.2, and $\mathrm{S}_{*\mathrm{I}}$ represent the set of $|\mathrm{S}_{*\mathrm{I}}| \leq |\mathrm{S}_*|$ contour regions after the integrated intensity/disparity grouping, as in Sect. 6.3.1. All contour regions are sorted by their horizontal positions. Disparity verification based contour grouping is performed on each region of $\{\mathrm{R} \in \mathrm{S}_* \mid \boldsymbol{\mu}_{\mathrm{D}}(\mathrm{R}) \neq 0\}$ by a recursive function **group()**, explained in Fig. 7–6 (ignoring boundary cases), and the grouping result, stored in $\mathrm{S}_{*\mathrm{D}}$, is dynamically updated; initially $\mathrm{S}_{*\mathrm{D}} = \mathrm{S}_*$. S_* is also updated to confirm the grouping results in $\mathrm{S}_{*\mathrm{I}}$. In Fig. 7–6, $\ltimes(\mathrm{R})$ and $\rtimes(\mathrm{R})$ are the left and right neighbors of R according to the sorted horizontal positions, respectively, and $\mathrm{R}_{\bowtie}$ the grouping candidate of R; initially $\mathrm{R}_{\bowtie} = \mathrm{R}$.

The algorithm proceeds as follows. First select the left or right neighbor of R to be $\mathrm{R}_{\bowtie}$, according to $\boldsymbol{\mu}_{\mathrm{D}}(\mathrm{R})$. If R and $\mathrm{R}_{\bowtie}$ have opposite grouping directions, then check if they overlap. If they do, group them if they match, and apply **group()** to any non-blocked part of R again; otherwise, skip $\mathrm{R}_{\bowtie}$, get the next horizontal neighbor as $\mathrm{R}_{\bowtie}$, and apply **group()**. If R and $\mathrm{R}_{\bowtie}$ have nondistinct grouping directions, then check if they are already grouped in $\mathrm{S}_{*\mathrm{I}}$. If they are, integrate $\mathrm{R}_{\bowtie}$ with R and retry **group()** on R; otherwise check if they overlap. If they do, group them if $\mathrm{R}_{\bowtie}$'s direction is unknown and R and $\mathrm{R}_{\bowtie}$ match, and apply **group()** to any non-blocked part of R; otherwise, skip $\mathrm{R}_{\bowtie}$, get the next horizontal neighbor to be $\mathrm{R}_{\bowtie}$, and reapply **group()**. Currently, $\tau_{\mathrm{I}} = 0.2$ and $\tau_{\mathrm{M}} = 20$ are empirical choices, but once set, no further parameter tuning was required during testing, as thresholding is performed on normalized values. The choice of $\tau_{\bowtie} = 50$ has proved non-critical.

group (**in** R, **in** $R_{\bowtie}$, **inout** S_*, **in** S_{*I}, **inout** S_{*D})

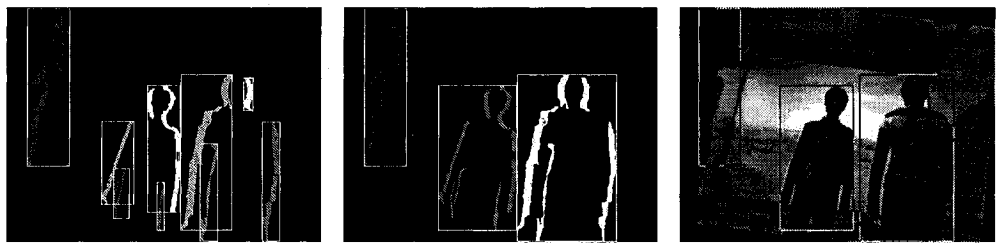
$$R_{\bowtie} \Leftarrow \begin{cases} \rtimes(R_{\bowtie}) \in S_*, & \textbf{if } \mu_D(R) < 0 \quad // \textit{ if left boundary, get right neighbor} \\ \ltimes(R_{\bowtie}) \in S_*, & \textbf{if } \mu_D(R) > 0 \quad // \textit{ if right boundary, get left neighbor} \end{cases}$$

if $\mu_D(R)\mu_D(R_{\bowtie}) < 0$ // *opposite grouping directions*
  if $\mu_{\natural}(R, R_{\bowtie}) > \tau_{\natural}$ // *overlap*
    if $\mu_M(R, R_{\bowtie}) < \tau_M$ // *match*
      group R, $R_{\bowtie}$ in S_{*D}
    if remains (R, $R_{\bowtie}$) = **true**
      group (R, $R_{\bowtie}$, S_*, S_{*I}, S_{*D})
    else done.
  else // R, $R_{\bowtie}$ *do not overlap*
    group (R, $R_{\bowtie}$, S_*, S_{*I}, S_{*D})
else // R, $R_{\bowtie}$ *have nondistinct grouping direction*
  if R, $R_{\bowtie}$ grouped in S_{*I}
    $R \Leftarrow R \cup R_{\bowtie}$
    update S_*, S_{*D}
    group (R, R, S_*, S_{*I}, S_{*D})
  else // R, $R_{\bowtie}$ *not grouped in* S_{*I}
    if $\mu_{\natural}(R, R_{\bowtie}) > \tau_{\natural}$ // *overlap*
      if $\mu_D(R_{\bowtie}) = 0 \wedge \mu_M(R, R_{\bowtie}) < \tau_M$ // $R_{\bowtie}$ *direction unknown and* R, $R_{\bowtie}$ *match*
        group R, $R_{\bowtie}$ in S_{*D}
      if remains (R, $R_{\bowtie}$) = **true** // R, $R_{\bowtie}$ *same direction or* R, $R_{\bowtie}$ *do not match*
        group (R, $R_{\bowtie}$, S_*, S_{*I}, S_{*D})
      else done.
    else // R, $R_{\bowtie}$ *do not overlap*
      group (R, $R_{\bowtie}$, S_*, S_{*I}, S_{*D})

Boolean remains (**inout** R, **in** $R_{\bowtie}$)

if $\mu_{\natural}(R, R_{\bowtie}) > 1 - \tau_{\natural}$ // R *blocked by* $R_{\bowtie}$
  return(**false**)
else // R *not blocked by* $R_{\bowtie}$
  $R \Leftarrow R - \pi_y^{-1}(\pi_y(R_i) \cap \pi_y(R_j))$ // *keep non-blocked part*
  return(**true**)

Figure 7–6: Contour grouping algorithm based on disparity verification.



(a) (b) (c)

Figure 7–7: Disparity contour grouping, left view. (a) Neighboring contour regions grouped based on integrated intensity-disparity measure μ_{I} as in Fig. 6–4(a). (b) Contour regions grouped based on disparity verification. (c) Objects located according to grouped contours.

Disparity contour grouping produces object regions as shown in Fig. 7–7(b). An object is considered a false one and removed if $\mu_{\text{M}}(\text{O}) \geq \tau_{\text{M}}$. Fig. 7–7(c) shows the segmentation result on the original image.

7.6 Experimental results

The proposed method was again validated using the same two image sequences as in Sect. 5.3 and Sect. 6.4, with sample images shown in Fig. 7–8 and Fig. 7–9 and the quantitative analysis illustrated in Table 7–1, Table 7–2, Fig. 7–10 and Fig. 7–11.

As can be seen, the proportion of 'correct' segmentation reaches about 85% in both sequences. In particular, the proportion of 'accurate' segmentation has significantly increased to between 55% and 60%. This is due to the advantage of disparity verification based grouping, which is based on disparity, i.e., depth information, and is much more robust to noise from background and object internal texture, as observable in frames 320∼370 of Fig. 7–9.

What is more, disparity verification segmentation demonstrates its improved capability in handling partial occlusions. In comparing the partial occlusion cases of Table 7–1 and Table 7–2, the proportion of 'accurate' segmentation has doubled over that of contour grouping segmentation. The results are also observable in frames 330 and 340 of Fig. 7–8 and frames 270∼285 of Fig. 7–9.

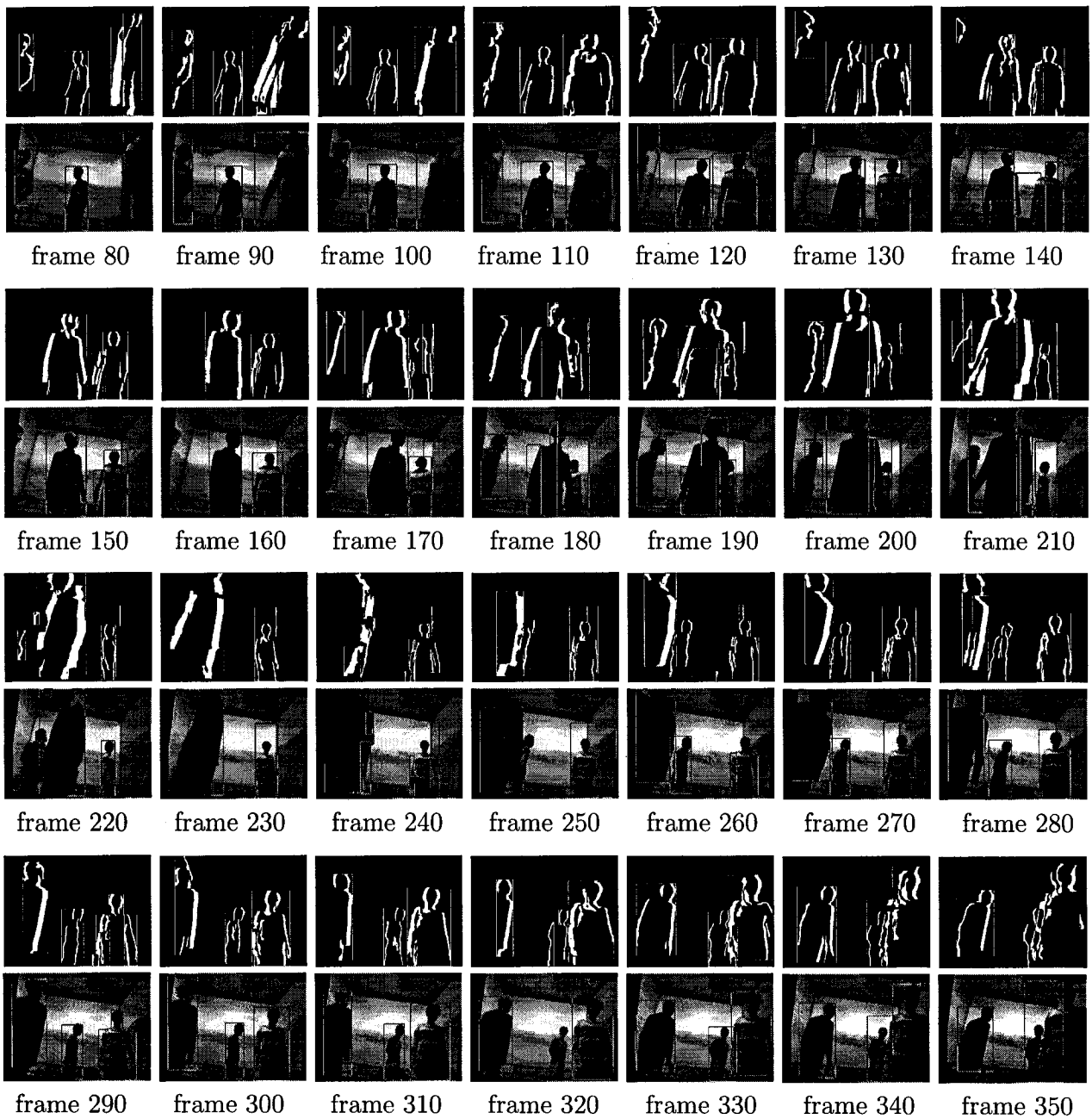


Figure 7–8: Disparity verification segmentation results on a sequence with static background and three subjects.

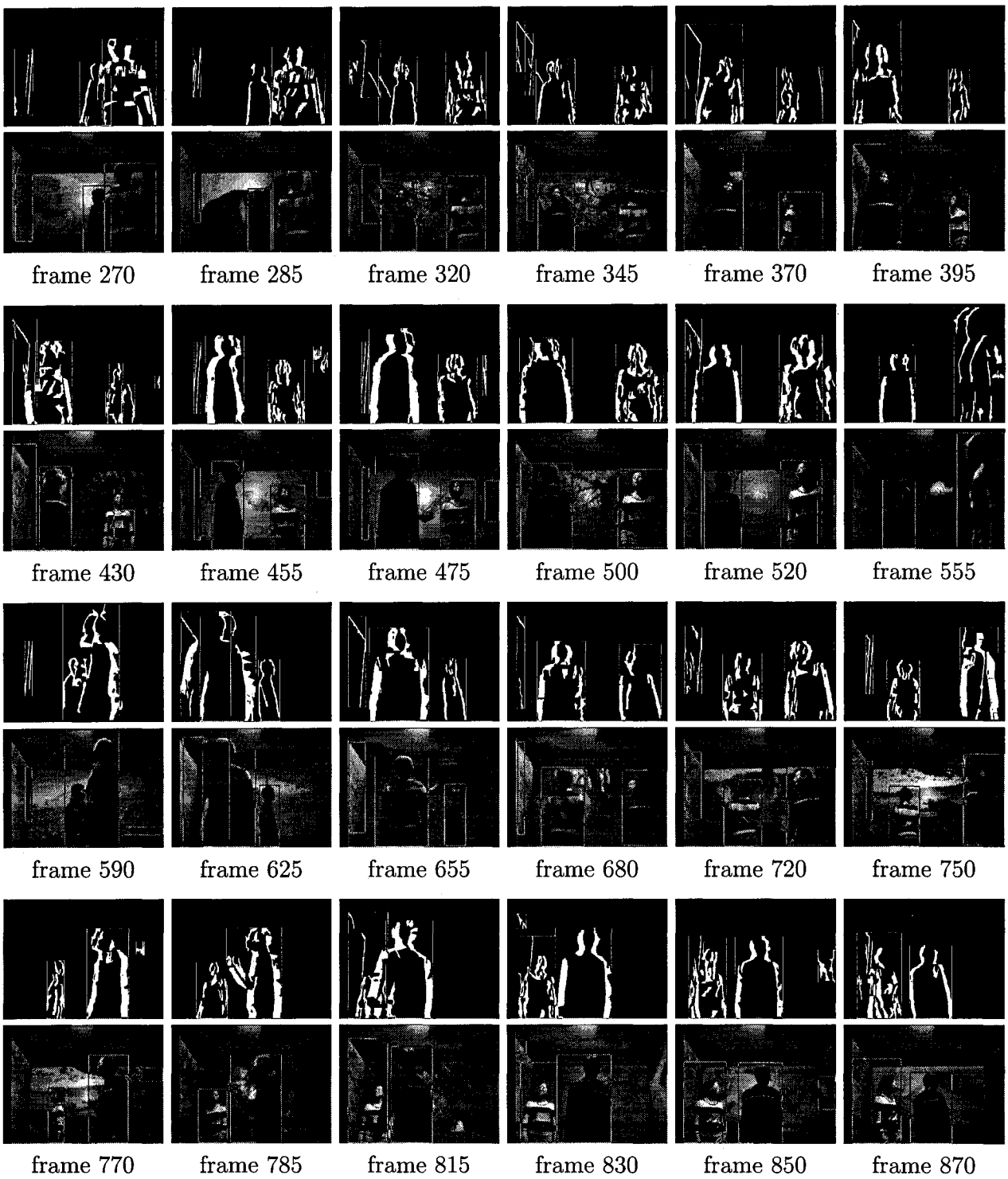


Figure 7–9: Disparity verification segmentation results on a sequence with moving video background and two subjects.

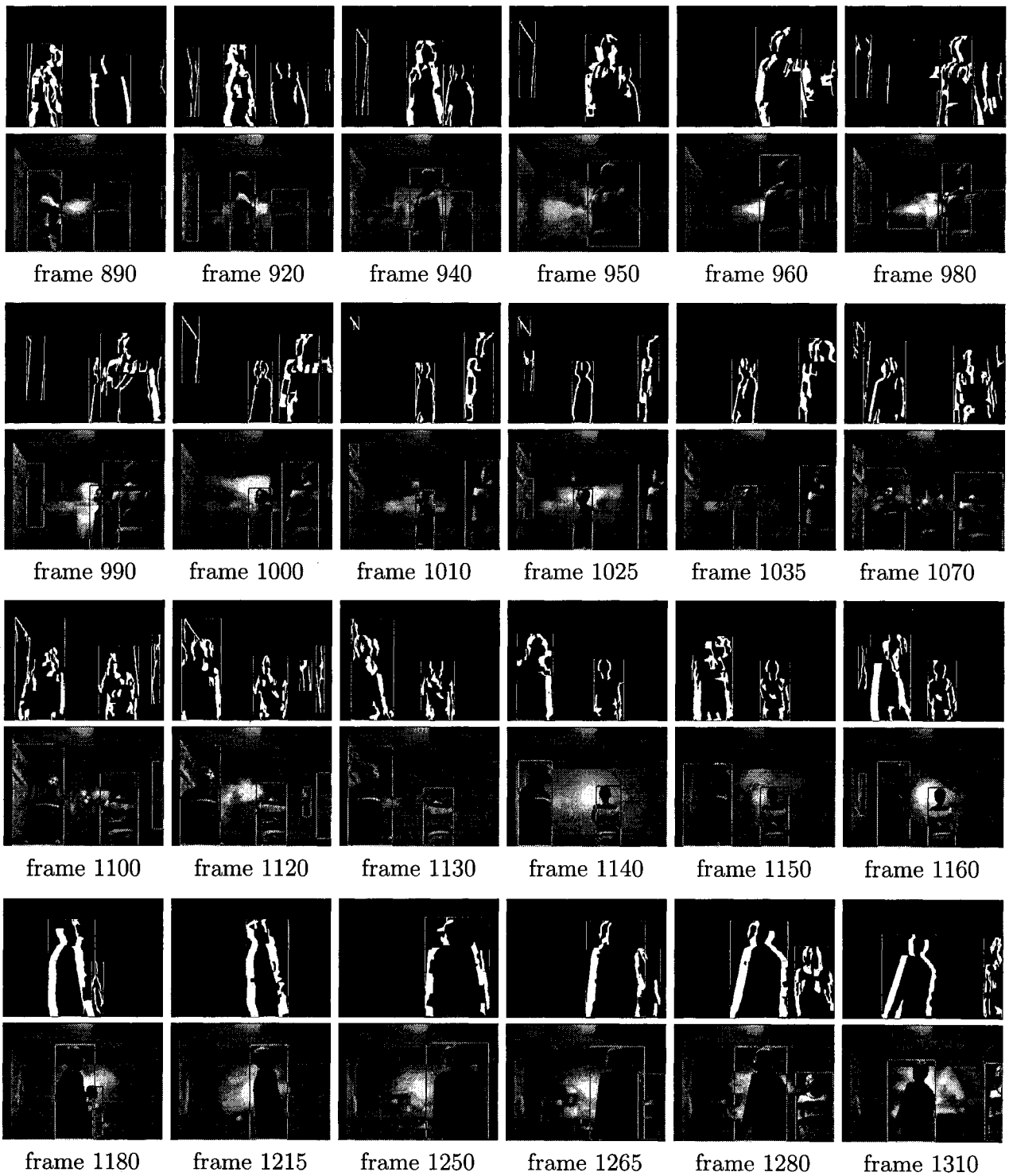


Figure 7–9: Disparity verification segmentation results on a sequence with moving video background and two subjects. (cont.)

Table 7–1: Segmentation result comparison on the static background sequence.

method	total true objects			segmentation accuracy: accurate: exact bdg. box		inaccurate: enlarged bdg. box		inaccurate: partial object		incorrect: enlarged ptl. obj.		incorrect: coalesced object		incorrect: object undet.		incorrect: false object	
disparity	no occl'n	591	100%	20	3.38%	95	16.07%	185	31.30%	20	3.38%	226	38.24%	45	7.61%		
multihistogram	ptl. occl'n	211	100%	18	8.53%	11	5.21%	56	26.54%	11	5.21%	112	53.08%	3	1.42%		
segmentation	all	802	100%	38	4.74%	106	13.22%	241	30.05%	31	3.87%	338	42.14%	48	5.99%	8	1.00%
contour	no occl'n	591	100%	306	51.78%	29	4.91%	123	20.81%	41	6.94%	78	13.20%	14	2.37%		
grouping	ptl. occl'n	211	100%	36	17.06%	3	1.42%	66	31.28%	1	0.47%	98	46.45%	7	3.32%		
segmentation	all	802	100%	342	42.64%	32	3.99%	189	23.57%	42	5.24%	176	21.95%	21	2.62%	40	4.99%
disparity	no occl'n	591	100%	376	63.62%	41	6.94%	112	18.95%	33	5.58%	16	2.71%	13	2.20%		
verification	ptl. occl'n	211	100%	66	31.28%	6	2.84%	69	32.70%	3	1.42%	61	28.91%	6	2.84%		
segmentation	all	802	100%	442	55.11%	47	5.86%	181	22.57%	36	4.49%	77	9.60%	19	2.37%	42	5.24%

Table 7–2: Segmentation result comparison on the video background sequence.

method	total true objects			segmentation accuracy: accurate: exact bdg. box		inaccurate: enlarged bdg. box		inaccurate: partial object		incorrect: enlarged ptl. obj.		incorrect: coalesced object		incorrect: object undet.		incorrect: false object	
disparity	no occl'n	1767	100%	317	17.94%	116	6.56%	926	52.41%	215	12.17%	192	10.87%	1	0.06%		
multihistogram	ptl. occl'n	334	100%	21	6.29%	10	2.99%	50	14.97%	1	0.30%	247	73.95%	5	1.50%		
segmentation	all	2101	100%	338	16.09%	126	6.00%	976	46.45%	216	10.28%	439	20.89%	6	0.29%	517	24.61%
contour	no occl'n	1767	100%	825	46.69%	149	8.43%	341	19.30%	177	10.02%	275	15.56%	0	0.00%		
grouping	ptl. occl'n	334	100%	63	18.86%	9	2.69%	48	14.37%	7	2.10%	203	60.78%	4	1.20%		
segmentation	all	2101	100%	888	42.27%	158	7.52%	389	18.51%	184	8.76%	478	22.75%	4	0.19%	817	38.89%
disparity	no occl'n	1767	100%	1113	62.99%	335	18.96%	182	10.30%	99	5.60%	38	2.15%	0	0.00%		
verification	ptl. occl'n	334	100%	127	38.02%	19	5.69%	38	11.38%	6	1.80%	140	41.92%	4	1.20%		
segmentation	all	2101	100%	1240	59.02%	354	16.85%	220	10.47%	105	5.00%	178	8.47%	4	0.19%	866	41.22%

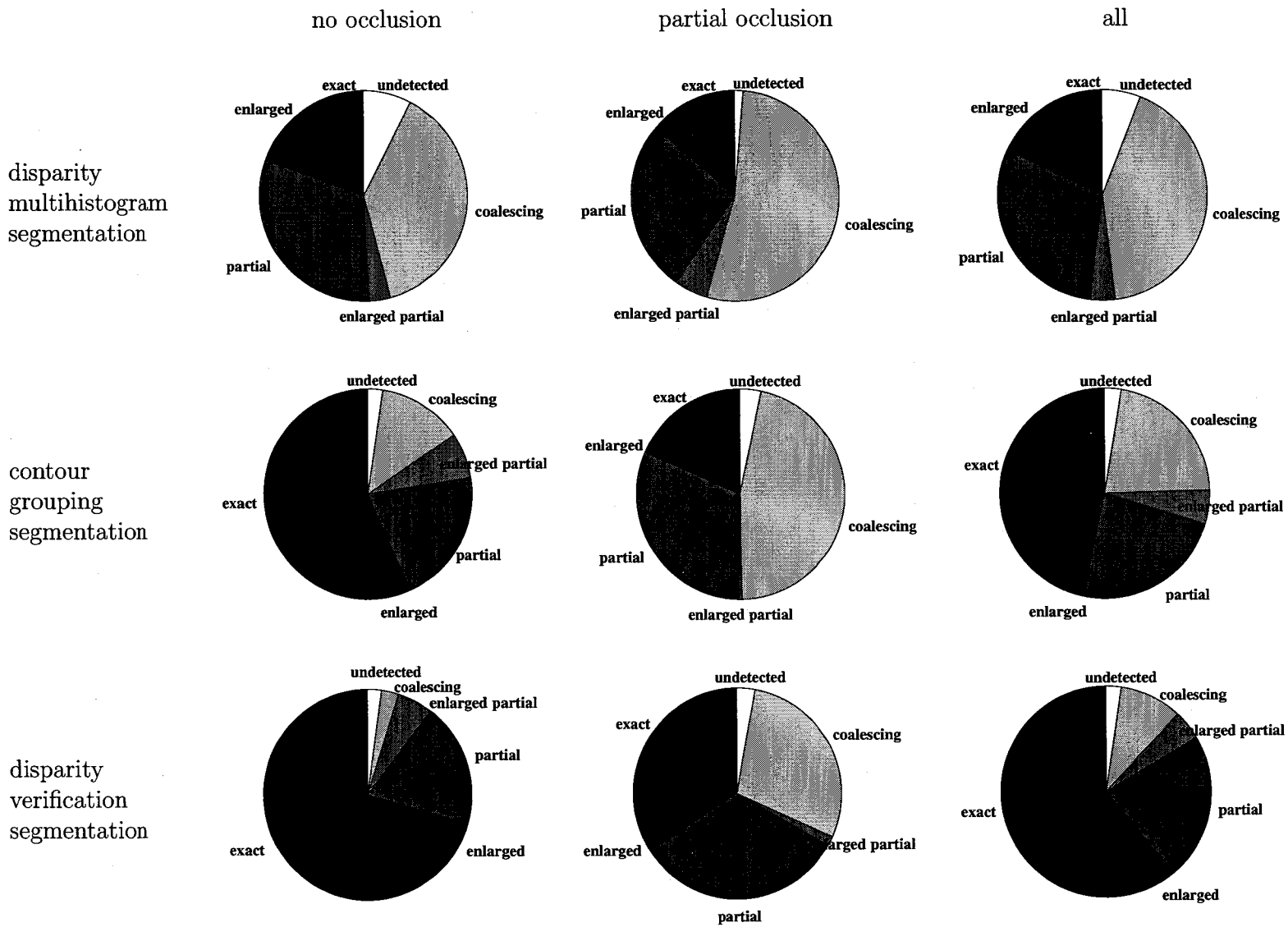


Figure 7–10: Segmentation result comparison on the static background sequence.

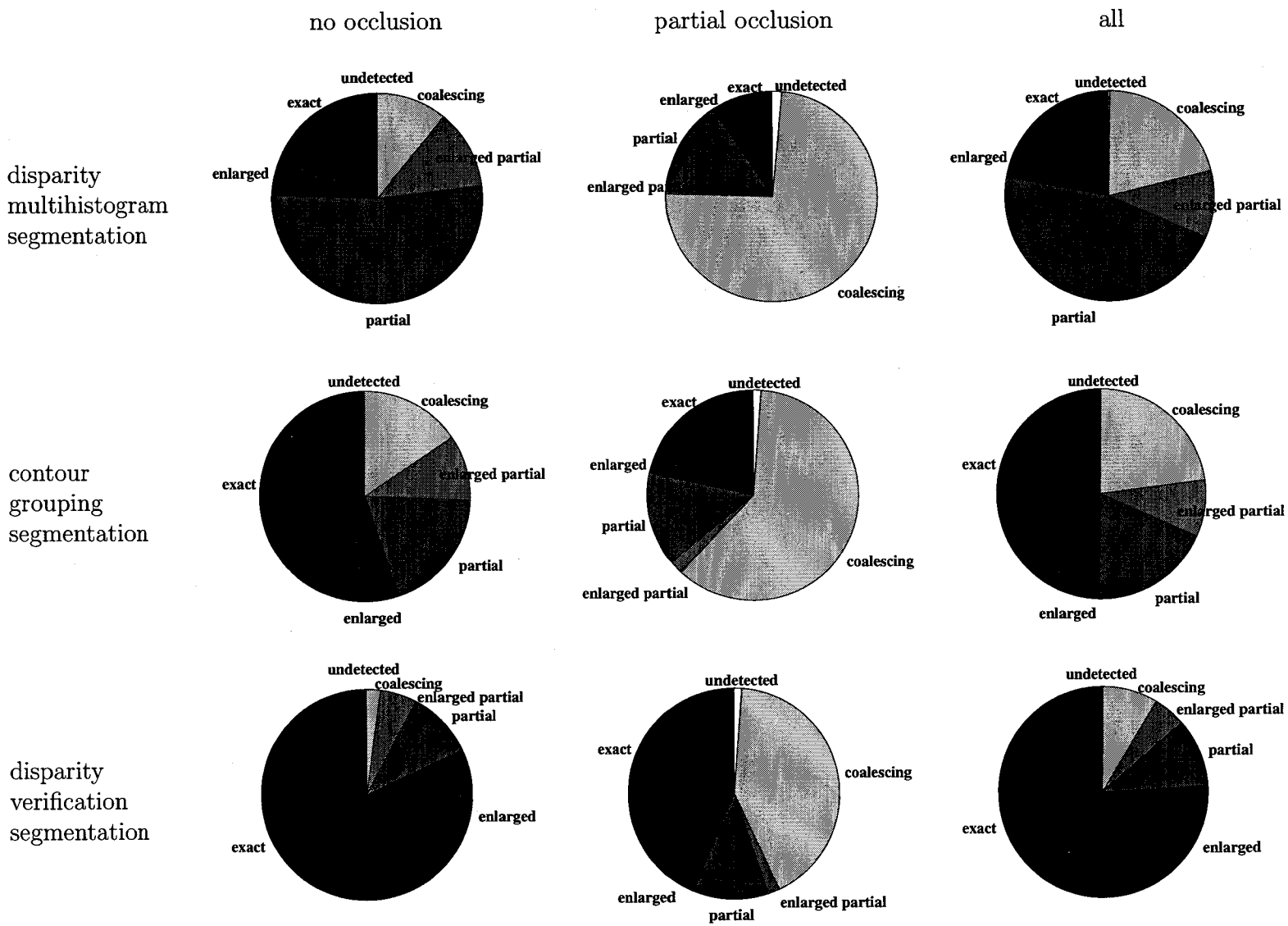


Figure 7–11: Segmentation result comparison on the video background sequence.

The biggest improvement of the disparity verification approach over the contour grouping approach is the removal of the convexity constraint on object shape, which improves the segmentation results from two directions. First, grouping difficulties due to the concave profile of an object or the wrong convexity measure of a contour are largely eliminated, as exemplified by frames 80 and 290~310 of Fig. 7–8 and frames 455, 655, 720, 785 and 1010 of Fig. 7–9 in comparison with the same frames in Fig. 6–6 and Fig. 6–7. Second, false groupings of multiple objects, for example as seen in frame 280 of Fig. 6–6 and frames 750, 770 and 830 of Fig. 6–7, are corrected. Therefore, 'object coalescing' decreases by over 50% compared to contour grouping segmentation, as we notice in Fig. 7–10 and Fig. 7–11.

Despite numerous improvements over the previous methods, the results of disparity verification segmentation are still not perfect. The first and the most important source of remaining error is the calibration of the environment. As mentioned in Sect. 4.3, the *background disparity map* (BDM) was constructed based on measurements of the environment with a tape measure. This results in inaccuracies that cause systematic background noise, confusing the segmenter, as visible on the left screen of the video background sequence in Fig. 7–9 with obvious examples shown in frames 270~475. This also explains the large number of false objects in the result.

Secondly, the disparity contour direction measure, $\mu_{\mathrm{D}}(\mathrm{R})$, of a contour stripe can provide misleading information due to texturelessness in both background and foreground, as explained in Fig. 7–3. A typical example can be found in frame 140 of Fig. 7–8 where the arms of two people are grouped together.

Third, without the aid of temporal coherence between consecutive frames, partial occlusion remains a difficult issue for a segmenter. This scenario results in an 'accurate' segmentation rate of only 1/3, with samples shown in frames 180~210 of Fig. 7–8 and frames 590 and 950 of Fig. 7–9.

Moreover, specular lighting caused by the directional illumination of the projector, as shown in frame 980 of Fig. 7–9, differences in camera response due to either sensor noise or sensor response mismatch, and camera synchronization error during synchronized video acquisition can

Table 7–3: Result analysis of disparity verification segmentation.

improvements over contour grouping approach	example frames: static background Fig. 6–6 vs. Fig. 7–8	example frames: video background Fig. 6–7 vs. Fig. 7–9
no convexity constraint	80, 290~310	455, 655, 720, 785, 1010
improved capability with occlusion	330, 340	270, 285
more robust to background noise and internal object texture		320~370
correctly judge contour grouping completion and reduce coalescing	280	750, 770, 830
remaining problems	**example frames: static background Fig. 7–8**	**example frames: video background Fig. 7–9**
background noise due to environment calibration error		270~475
wrong contour direction due to textureless foreground/background	140	
difficulty with partial occlusion	180~210	590, 950
directional illumination of projector		980

all contribute to differences in image intensity values between corresponding pixels in the left and right views. The camera synchronization problem is exacerbated by the interlaced NTSC cameras used for experimentation, which produce a 640 × 480 image frame by interlacing two 640 × 240 fields. This interlacing effect is further amplified by image distortion removal and rectification, during preprocessing.

A summary of the improvements and problems discussed above is shown in Table 7–3.

7.7 Performance analysis

The initial, unoptimized, implementation of the algorithm presented processes about four 640 × 480 monochrome image pairs per second on a 1.8GHz AMD processor running in 32 bit mode. Informal analysis of the algorithm suggests that potential performance is significantly better, comparable to the cost of a few linear passes over the input data. Crucially, the construction of BDM takes place offline and does not contribute to the cost.

In an optimized implementation, all pixel-level processing described can be performed in two passes. The first of these computes Eq. (4.3) from a pair of input images. To the extent

that the BDM is vertically homogeneous, which is characteristic of large areas of indoor scenes, a rolling implementation annuls the overhead of the vertical stripe matching window.

The second pixel-grain pass applies the edge detection operator and computes conditions 1 and 2 of Eq. (6.1) for both high and low thresholds, producing a queue of edge candidates for each scanline in the rectified view. These points are combined into line segments marked as to which threshold they satisfy, and fed to a region-growing algorithm. This builds a higher level representation of the region list, computing the statistics of Eq. (6.3)-(6.8) and (6.11)-(6.13) and propagating the overlap condition of Eq. (6.2) on the fly. Pairing candidate edge points into segments is in principle quadratic in the length of the scanlines, but the three conditions of Eq. (6.1) constrain the search space in such a way that the practical cost of this operation is comparable to that of an additional pixel scan of the full scene.

The regions can now be scanned to regularize line segment outliers by Eq. (6.10), and to determine R^*. The outlier regions can then be culled according to Eq. (6.14) and grouped based on integrated intensity/disparity in a single pass by Eq. (6.15)-(6.18). Again the operation is in principle quadratic, this time in the number of surviving regions, but the reduction in data volume to this point is such that in practice the number of operations is no more than 10 times the number of total objects.

The final disparity verification based contour grouping by Eq. (7.2)-(7.11) and Fig. 7–6 is dominated by the computations of foreground confidence in Eq. (7.7) and of matching cost between regions in Eq. (7.10). In the worst case, Eq. (7.7) and Eq. (7.10) each requires two linear passes of pixel-level processing of the whole image.

7.8 Comparison with graph cut method

The presented disparity verification segmentation is compared to one of the best stereo methods—Kolmogorov and Zabih's graph cut algorithm [93], discussed in Sect. 2.2.3. Using the same camera calibration data as described in Chapter 3, their software [90] was tested on samples from the same sequences as shown in Fig. 7–12(a) and Fig. 7–13(a). Fig. 7–12(b)(c) and Fig. 7–13(b)(c) show the scene disparity map resulting from the graph cut algorithm on these

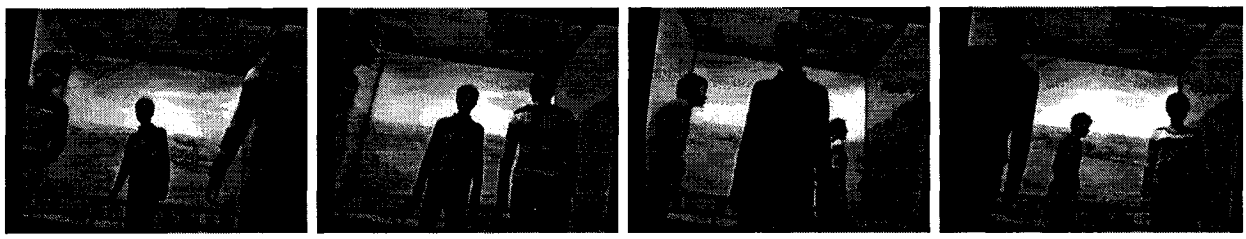
Table 7–4: Performance comparison of disparity contour approach and graph cut method. Image resolution 640 × 480. Processor: 1.8GHz 32 bit AMD (≈ 1.8GHz Pentium III).

method	processing time per frame (s)	frame rate (Hz)
disparity multihistogram	0.219	4.56
contour grouping	0.258	3.88
disparity verification	0.264	3.79
graph cut with occlusion [92] (Kolmogorov and Zabih 2001)	441.6	0.00229

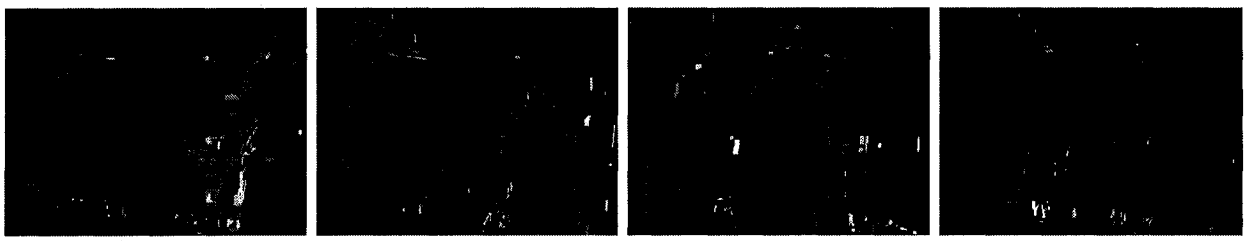
sample images. Fig. 7–12(d)(e) and Fig. 7–13(d)(e) show the results of disparity verification segmentation for comparison.

We can see that the graph cut algorithm offers reasonably good performance in generating a disparity map from a given scene despite the dynamic lighting and texture of the background. However, due to its weakness in dealing with large textureless regions, quite prevalent in the projected background, the resulting disparity map is inaccurate. This is especially evident in the results of the static background sequence where disparity values in the background area (Fig. 7–12(b)) are often incorrect. Object segmentation based on this result remains a challenge and a large amount of post-processing will be necessary.

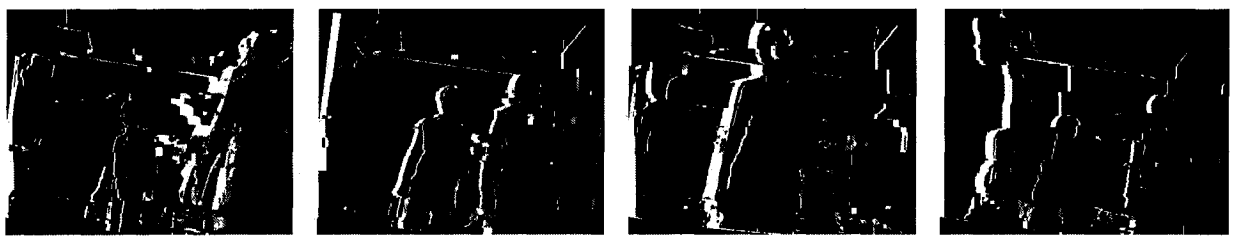
Table 7–4 compares the processing time of the proposed disparity contour based approach to Kolmogorov and Zabih's graph cut method. It can be seen that the graph cut algorithm, although only generating a disparity map without performing the entire object segmentation task, takes more than 7min for processing each frame pair, more than 1600 times slower than the proposed disparity contour based methods. Obviously, this is impractical for real-time, interactive applications.



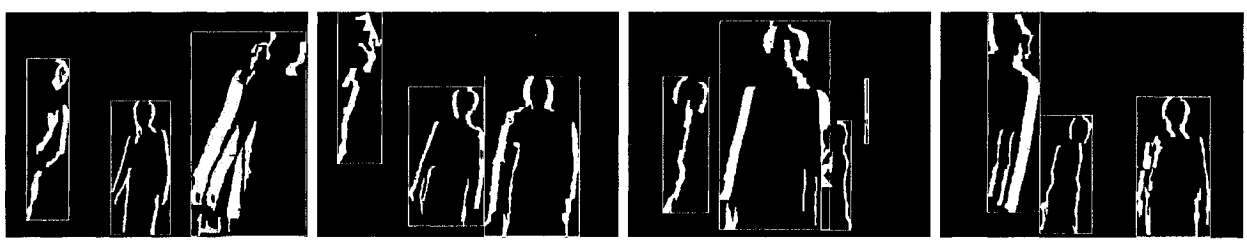
(a) Sample images after distortion removal and rectification.



(b) Scene disparity map by graph cut algorithm.



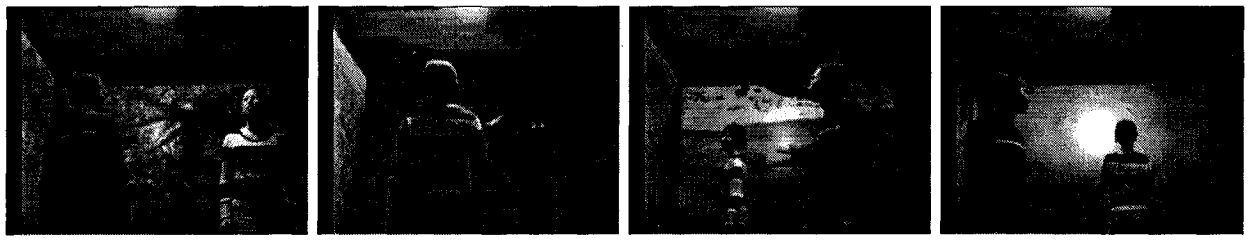
(c) Same results in (b) with occluded pixels indicated by white.



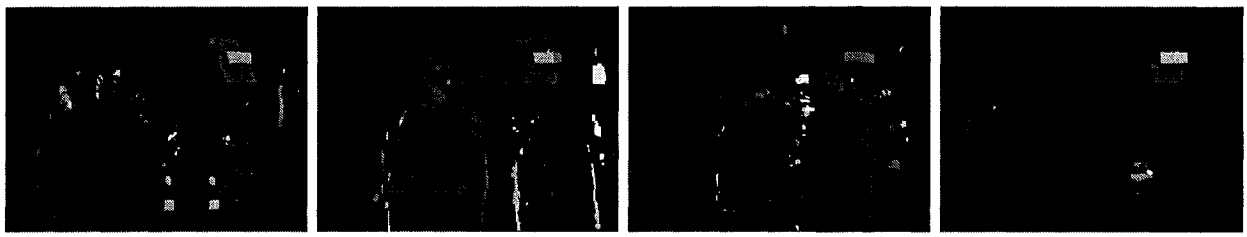
(d) Results by disparity verification based contour grouping.

(e) Final results of disparity verification segmentation.

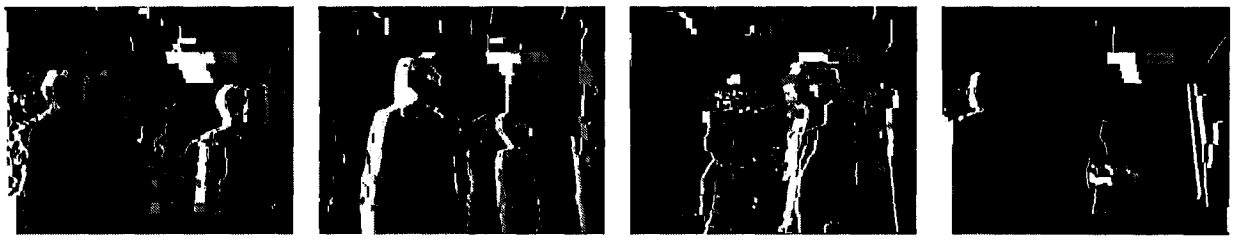
Figure 7–12: Comparison with Kolmogorov and Zabih's graph cut algorithm on static background sequence, left view.



(a) Sample images after distortion removal and rectification.

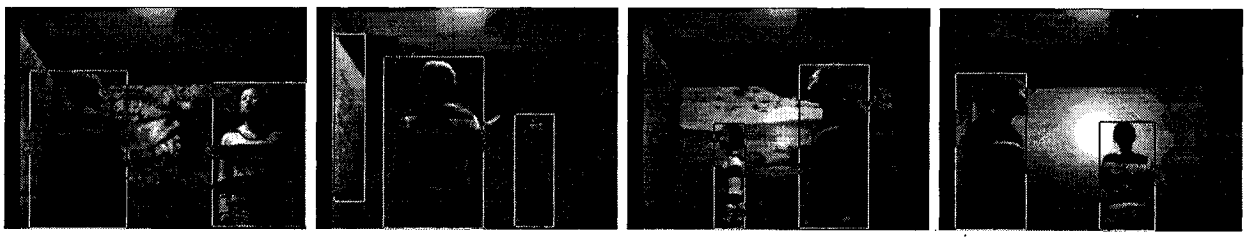


(b) Scene disparity map by graph cut algorithm.



(c) Same results in (b) with occluded pixels indicated by white.

(d) Results by disparity verification based contour grouping.



(e) Final results of disparity verification segmentation.

Figure 7–13: Comparison with Kolmogorov and Zabih's graph cut algorithm on video background sequence, left view.

CHAPTER 8
Conclusions and Future Work

This chapter summarizes the work presented in this thesis and suggests a number of improvements for the future.

8.1 Conclusions

A new multiple object segmentation system using disparity contours is developed. By exploiting the observation that background geometry may be more stable over time than texture or illumination, and avoiding full stereo reconstruction and empirical parameter tuning, the method easily achieves near-real-time performance on traditionally difficult scenes, without requiring identical or high-quality cameras, and is applicable to environments with dynamic lighting and texture.

Three different approaches developed chonologically during the study have been compared with deeper understanding achieved at each step. The disparity multihistogram approach, due to its naive noise removal and contour grouping strategy, yields systematically inaccurate segmentation results. The contour grouping approach, which employs a multi-evidential dual-threshold contour filter, statistical noise removal, and heuristic based contour grouping, overcomes the weaknesses of the multihistogram method and is far less dependent on empirical parameter tuning. However, its convexity assumption about object shape and its over-simplified calculation of the convexity measure limit the robustness of the contour grouping approach. By examining more closely the idea of disparity verification and exploring more features of disparity contours, the disparity verification based segmentation removes any constraint imposed on object shape and demonstrated its improved capability in handling partial occlusions. The disparity verification method achieves 85% 'correct' segmentation with approximately 60% 'accurate' segmentation in the experiments at a speed of nearly 4 frames/sec on videos of 640×480 pixels.

8.2 Future work

Although as noted in Chapter 7 there remain a number of challenges due to errors caused by environment calibration (which could be improved by using special measurement assembly such as a laser scanner), incorrect disparity contour direction, and partly occluded boundaries, the method has not yet been exploited to its full potential.

8.2.1 Boundary finder

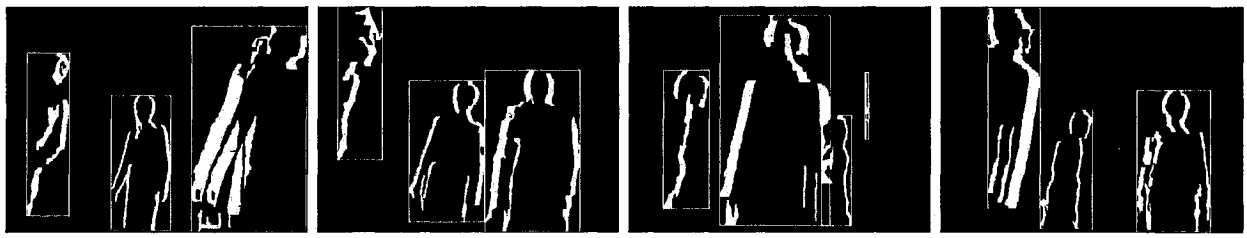
As explained in Section 4.2, an object contour lies on the left side of the object boundary in the left image and on the right side of the boundary in the right image. Therefore, precise boundaries can be located by connecting either the negative edge points on the object's peripheral contour in the left view or the positive edge points in the right view, once the object contours are correctly grouped.

Fig. 8–1 and Fig. 8–2 show objects cropped from a scene based on boundaries located from peripheral contours. We can see that although the segmentation is correct overall, the accuracy of boundary location is poor.

A major cause of such inaccuracies is that the view difference images, from which disparity contours are extracted, are the result of an intensity based block matching technique (Eq. (4.3)). The detection of peripheral contours relies on the local intensity difference between background and foreground. When such a difference is truly small in some localities, no disparity contour can be detected there, and thus no peripheral contour is obtained. The internal contours resulting from foreground auto-correlation are then mistaken by the boundary finder for the peripheral contours, and therefore object boundaries are mislocated. This is especially severe in the video background sequence (see Fig. 8–2) where the dynamic background texture increases the occurrence of local intensity proximity between background and foreground.

The second problem, due to the inherent weakness of the horizontally positioned stereo system, is the failure to detect horizontal or near horizontal disparity contours of an object. Extrapolation to create horizontal or near horizontal boundaries based on detected vertical or

(a) Sample images overlaid with object bounding boxes.



(b) Results of disparity verification based contour grouping.

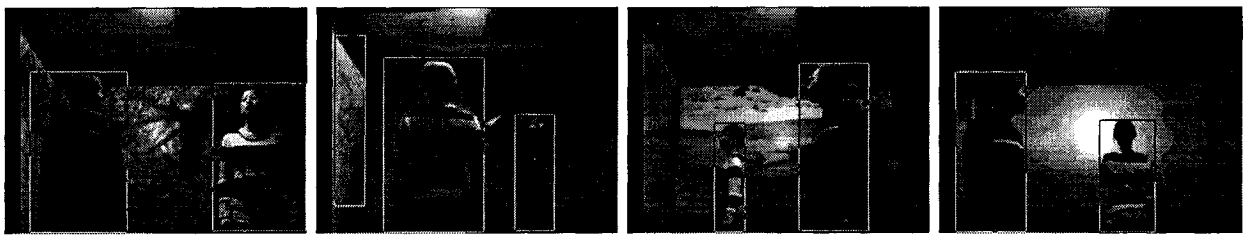
(c) Objects segmented by boundaries located from peripheral contours.

Figure 8–1: Results of object boundary finder on static background sequence, left view.

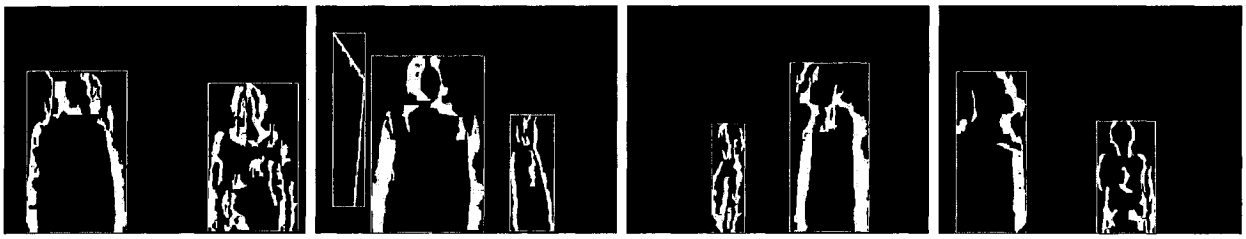
near vertical contours is prone to error. As observable in the last two samples of Fig. 8–1, the shoulder of the person on the left is mislocated.

In the presence of partial occlusions or when an object is partially out of view, the peripheral contours are either invisible, or of irregular shape, which makes boundary location difficult. Self-occlusion, such as the arm of the person on the right in the first sample of Fig. 8–1, is also an issue requiring special attention in the boundary finder.

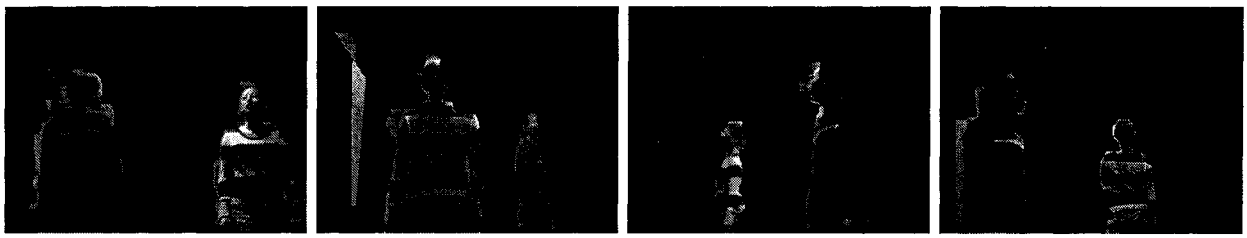
In summary, a better boundary finder is necessary to overcome or compensate for the above shortcomings and produce a more accurate object location, which will provide potential for applications that require more details of body motion such as gesture recognition [117].



(a) Sample images overlaid with object bounding boxes.



(b) Results of disparity verification based contour grouping.



(c) Objects segmented by boundaries located from peripheral contours.

Figure 8–2: Results of object boundary finder on video background sequence, left view.

8.2.2 2D and 3D tracking

By exploiting the temporal coherence in an image sequence, a higher-level tracker should alleviate many problems due to contour extraction error and partial occlusion. The extended system is illustrated in Fig. 8–3. Here two new modules are added, object prediction and object refinement.

The final segmentation result, combined with extracted raw contour information, is fed to the prediction module, which predicts the location of contours as well as objects. The prediction is then verified against the initial segmentation result to refine object information. This feedback loop is expected to propagate good segmentation results and deal with occlusions by keeping track of contours.

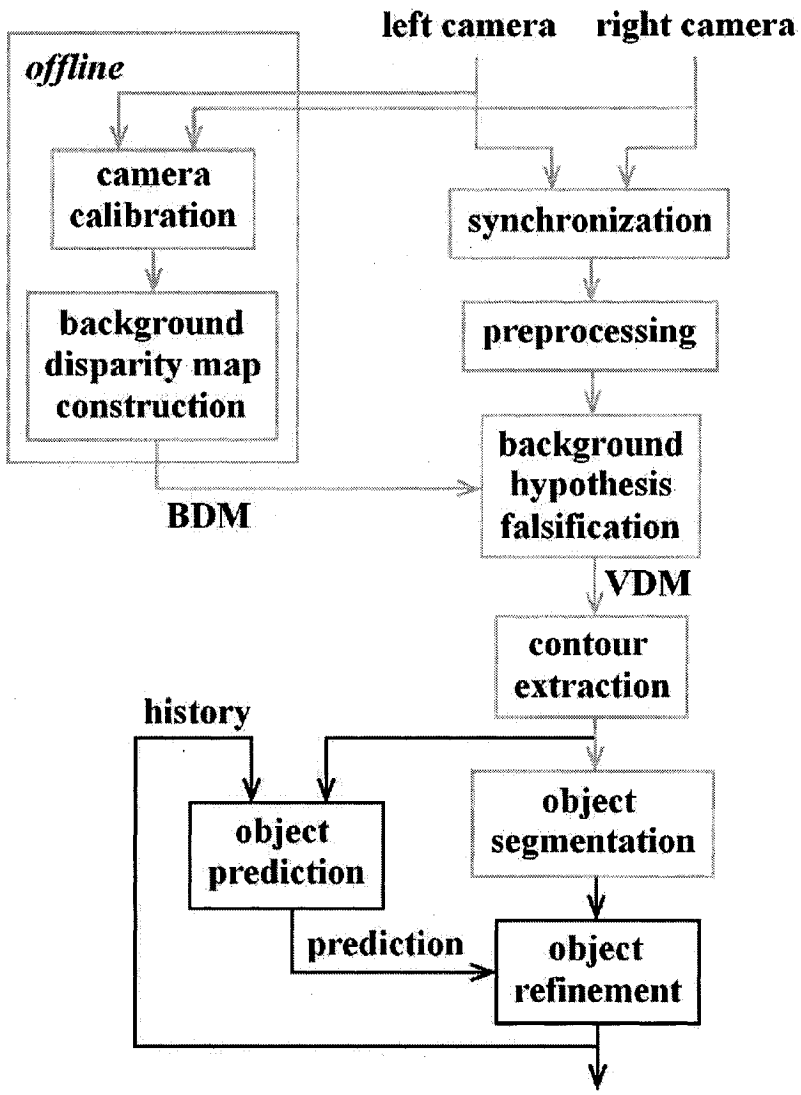


Figure 8–3: A disparity-based object tracking system.

In addition, enhancing the object finder with a pre-trained model of human (or other) target contours should greatly increase the precision and reliability of tracking [16] and reduce the problem of wrongly oriented contour regions.

The information exposed by the disparity contour technique is far from exhausted by object segmentation. Since absolute depth can easily be extracted, the method naturally extends to 3D object tracking.

8.2.3 Online update of background disparity map

The *background disparity map* (BDM) is presently assumed to be static and constructed offline, however, there is no reason why this could not be updated online to integrate changes to the environment and to correct the inaccuracies in the initial BDM caused by camera calibration error. This online update can be carried out by a parallel process with low priority, which not only has little impact on the system performance but also opens a door to more complicated and expensive algorithms. This improvement would make the method more suitable in environments in which the geometry of the background, and not just its texture, is dynamic somewhat.

8.3 Application to Shared Reality Environment

A number of vision tasks are involved in order to realize the goals of the SRE project, including background removal, person tracking, gesture recognition, gaze tracking, and arbitrary view synthesis. If using frontal video projection, shadow removal is also necessary. The object segmentation and potential tracking system presented in this thesis can either be applied directly to perform some of the tasks, such as background removal and person tracking, or aid in finding solutions to others, such as shadow removal and gesture recognition.

In addition to the previously suggested improvements in boundary finding, tracking and BDM update, a few other practical issues need to be resolved. As noted in Sect. 8.2.1, the inherent weakness of the horizontally positioned stereo system results in a failure to detect horizontal or near horizontal object boundaries, which is expected to cause difficulty in gesture recognition. This weakness can be overcome by employing three cameras positioned in a vertical "L" shape, combining the results of both horizontal and vertical sensing systems. In order to achieve online real-time performance for the SRE, a distributed processing system will eventually need to be used. In this case, the system will consist of multiple nodes, each with a camera and a processor. Images acquired from each camera will be processed locally and the information for stereo processing will be shared via the network. To compensate for the difference in the speed of capture cards and processors, video acquisition will need to be synchronized not only at the starting point, as explained in Sect. A.1, but also regularly online. Specifically for shadow

removal, geometric calibration of the projector as well as the camera will be necessary for the precise location of the occluder and the shadow.

The multi-camera object segmentation system presented is also useful for a wide variety of applications including surveillance, tracking, and event recognition. It is particularly interesting to virtual reality environment research and the entertainment and film industry. The proposed disparity contour method is fast, and immune to large textureless regions and rapid changes in background texture and illumination, difficulties so far untackled by existing approaches.

APPENDIX A
Video Acquisition and Preprocessing

This section briefly describes synchronized video acquisition using multiple cameras and preprocessing including distortion removal and rectification.

A.1 Synchronized video acquisition

Video acquisition from multiple cameras was carried out by a distributed system in which each camera was connected to a computer. Multicast synchronizing signals were used to initiate and terminate video capture, which, in theory, runs at a stable rate of 30 frames/sec.[1] Due to the various response times and processing latencies of hardware, the starting time of a captured video varies with different cameras and capture cards. Moreover, at the beginning of a captured image sequence, the frame rate is usually not constant at 30 Hz due to the video streaming setup of a frame grabber. Therefore, off-line processing is necessary to locate the starting synchronized frame in each image sequence before any subsequent image processing. This was done based on timestamps recorded with each frame during video capture.

A.2 Image distortion removal and rectification

Distortions in captured images due to camera lens systems must be removed because the following rectification algorithm assumes a linear camera model without distortion. The undistorted image samples are shown in Fig. A–1(b).

As mentioned in Sect. 2.2.2, stereo matching can be made more efficient by making use of the epipolar constraint, searching for a match along corresponding epipolar lines. For this purpose, image rectification based on the algorithm of Fusiello et al. [60] is employed to virtually change the camera parameters, or equivalently, to warp pairs of stereo images so that pairs of

[1] Actually, the frame rate of an NTSC camera is $30 \times 1000/1001 \approx 29.97$ frames/sec.

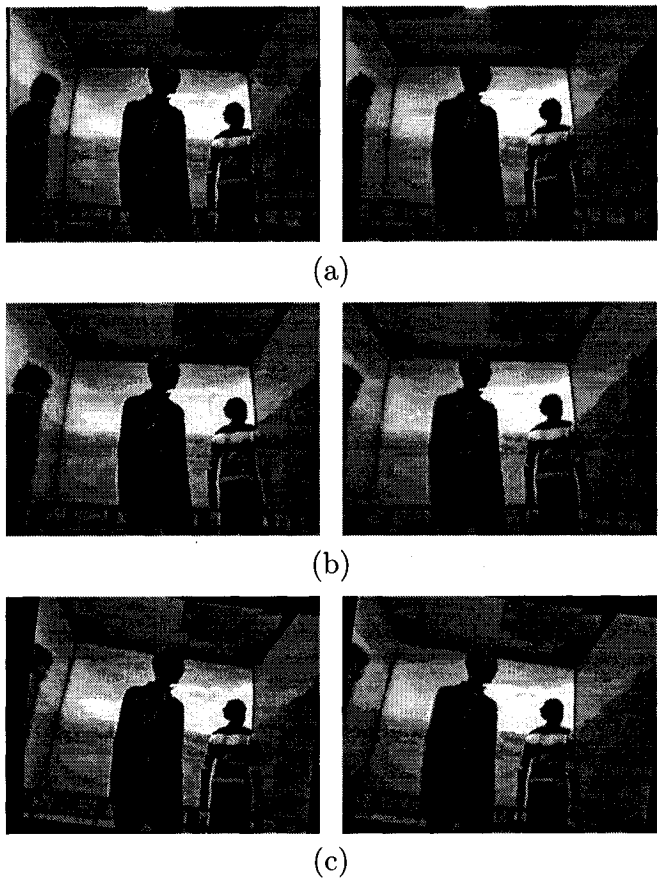


Figure A–1: Image preprocessing. Left column: left view; right column: right view. (a) Captured images. (b) Image distortions removed. (c) Images rectified.

conjugate epipolar lines become co-linear and parallel to the horizontal image axis. The results are shown in Fig. A–1(c).

Both distortion removal and rectification are image warping operations and can, therefore, be combined into a single lookup table that contains pixel-to-pixel mappings between the original image and the warped result. This lookup table can be built offline to reduce the run time cost.

References

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: a review. *Computer Vision and Image Understand*, 73(3):428–440, 1999.

[2] N. Atsushi, K. Hirokazu, H. Shinsaku, and I. Seiji. Tracking multiple people using distributed vision systems. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2974–2981, 2002.

[3] S. Ayer and H. S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *Int'l Conf. on Computer Vision*, pages 777–784, 1995.

[4] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *Int'l Joint Conf. on Artificial Intelligence*, pages 631–636, 1981.

[5] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Computer Vision and Pattern Recognition*, pages 434–441, 1998.

[6] H. Ballard and M. Brown. *Computer vision.* Prentice-Hall, New York, 1982.

[7] C. Beleznai, B. Frühstück, and H. Bischof. Human detection in groups using a fast mean shift procedure. In *Int'l Conf. on Image Processing*, volume 1, pages 349–352, 2004.

[8] P. N. Belhumeur. Bayesian approach to binocular stereopsis. *Int'l Journal of Computer Vision*, 19(3), 1996.

[9] J. R. Bergen, P. J. Burt, R. Hingorani, and S. Peleg. A three-frame algorithm for estimating two-component image motion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(9):886–896, 1992.

[10] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. In *Int'l Conf. on Computer Vision*, pages 1073–1080, 1998.

[11] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Int'l Conf. on Computer Vision*, pages 489–495, 1999.

[12] M. J. Black and D. J. Fleet. Probabilistic detection and tracking of motion discontinuities. In *Int'l Conf. on Computer Vision*, pages 551–558, 1999.

[13] M. J. Black, D. J. Fleet, and Y. Yacoob. A framework for modeling appearance change in image sequences. In *Int'l Conf. on Computer Vision*, pages 660–667, 1998.

[14] M. J. Black and A. D. Jepson. Eigentracking: robust matching and tracking of articulated objects using a view-based representation. In *European Conf. on Computer Vision*, pages 329–342, 1996.

[15] M. J. Black, Y. Yacoob, A. D. Jepson, and D. J. Fleet. Learning parameterized models of image motion. In *Computer Vision and Pattern Recognition*, pages 561–567, 1997.

[16] A. Blake and M. Isard. *Active contours*. Springer-Verlag London, 1998.

[17] S. Bougnoux. From projective to Euclidean space under any practical situation, a criticism of self-calibration. In *Int'l Conf. on Computer Vision*, pages 790–796, January 1998.

[18] J.-Y. Bouguet. Camera calibration toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc`, 2005.

[19] K. Boyer and A. Kak. Color-encoded structured light for rapid active ranging. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(1):14–28, 1987.

[20] K. Boyer and A. Kak. Structural stereopsis for 3-D vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(2):144–166, 1988.

[21] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Int'l Conf. on Computer Vision*, pages 105–112, 2001.

[22] Y. Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[23] Y. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[24] M. Brand. Shadow puppetry. In *Int'l Conf. on Computer Vision*, pages 1237–1244, 1999.

[25] D. C. Brockelbank and Y. H. Yang. An experimental investigation in the use of color in computational stereopsis. *IEEE Trans. Systems, Man, and Cybernetics*, 9(6):1365–1383, 1989.

[26] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(8):993–1007, 2003.

[27] F. Bunyak, I. Ersoy, and S. Subramanya. Shadow dectection by combined photometric invariants for improved foreground segmentation. In *IEEE Workshop on Applications of Computer Vision*, pages 510–515, 2005.

[28] P. Burt and B. Julesz. A disparity gradient limit for binocular vision. *Science*, 208:615–617, 1980.

[29] Q. Cai and J. K. Aggarwal. Automatic tracking of human motion in indoor scenes across multiple synchronized video streams. In *Int'l Conf. on Computer Vision*, pages 356–362, 1998.

[30] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 3(6):679–698, 1986.

[31] B. Caprile and V. Torre. Using vanishing points for camera calibration. *Int'l Journal of Computer Vision*, 4(2):127–140, 1990.

[32] T.-H. Chang and S. Gong. Tracking multiple people with a multi-camera system. In *IEEE Workshop on Multi-Object Tracking*, pages 19–26, 2001.

[33] C. Chatterjee and V. P. Roychowdhury. Algorithms for coplanar camera calibration. *Machine Vision and Applications*, 12(2):84–97, 2000.

[34] S.-C. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In *Proc. SPIE - Int'l Soc. for Optical Engineering, Visual Communications and Image Processing*, volume 5308, pages 881–892, 2004.

[35] K. Choo and D. J. Fleet. People tracking using hybrid Monte Carlo filtering. In *Int'l Conf. on Computer Vision*, volume II, pages 321–328, 2001.

[36] R. T. Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition*, volume 2, pages 234–240, 2003.

[37] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Int'l Conf. on Computer Vision*, pages 1197–1203, 1999.

[38] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[39] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* McGraw-Hill, New York, 1990.

[40] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. Maximum likelihood stereo algorithm. *Computer Vision and Image Understand*, 63(3):542–567, 1996.

[41] I. J. Cox, S. Roy, and S. L. Hingorani. Dynamic histogram warping of image pairs for constant image brightness. In *Int'l Conf. on Image Processing*, volume 2, pages 366–369, 1995.

[42] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, 2003.

[43] B. Curless and M. Levoy. Better optical triangulation through spacetime analysis. In *Int'l Conf. on Computer Vision*, pages 987–994, 1995.

[44] C. Davies and M. Nixon. A Hough transform for detecting the location and orientation of three-dimensional surfaces via color encoded spots. *IEEE Trans. Systems, Man, and Cybernetics*, 28(1B):90–95, 1998.

[45] L. S. Davis. Shape matching using relaxation techniques. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-1(1):60–72, 1979.

[46] U. R. Dhond and J. K. Aggarwal. Structure from stereo—a review. *IEEE Trans. Systems, Man, and Cybernetics*, 19(6), 1989.

[47] S. L. Dockstader and A. M. Tekalp. Multiple camera tracking of interacting and occluded human motion. *Proc. of the IEEE*, 89(10):1441–1455, 2001.

[48] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002.

[49] J. H. Elder and R. M. Goldberg. Ecological statistics of Gestalt laws for the perceptual organization of contours. *Journal of Vision*, 2:324–353, 2002.

[50] J. H. Elder, A. Krupnik, and L. A. Johnston. Contour grouping with prior models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(25):1–14, 2003.

[51] R. Fablet and M. J. Black. Automatic detection and tracking of human motion with a view-based representation. In *European Conf. on Computer Vision*, pages 476–491, 2002.

[52] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. *IEEE Trans. on Image Processing*, 7(3):336–344, 1998.

[53] O. Faugeras, T. Luong, and S. Maybank. Camera self-calibration: theory and experiments. In *European Conf. on Computer Vision*, pages 321–334, May 1992.

[54] O. Faugeras and G. Toscani. The calibration problem for stereo. In *Computer Vision and Pattern Recognition*, pages 15–20, June 1986.

[55] I. Feldmann, S. Askar, N. Brandenburg, P. Kauff, and O. Schreer. Real-time segmentation for advanced disparity estimation in immersive videoconference applications". In *Proc. of 10th Int. Conference on Computer Graphics, Visualization and Computer Vision (WSCG 2002)*, pages 171–178, 2002.

[56] D. J. Fleet, M. J. Black, and A. D. Jepson. Motion feature detection using steerable flow fields. In *Computer Vision and Pattern Recognition*, pages 274–281, 1998.

[57] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice in C.* Addison-Wesley, 2 edition, 1997.

[58] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: combining multi-image stereo and shading. *Int'l Journal of Computer Vision*, 16(1):35–56, 1995.

[59] A. Fusiello. Uncalibrated euclidean reconstruction: a review. *Image and Vision Computing*, 18:555–563, 2000.

[60] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[61] A. Gagalowicz and L. Vinet. Region matching for stereo pairs. In *Proc. 6th Scandinavian Conf. on Image Analysis*, pages 63–71, 1989.

[62] M. A. Gennert. Brightness-based stereo matching. In *Int'l Conf. on Computer Vision*, pages 139–143, 1988.

[63] W. E. L. Grimson. Computational experiments with a feature-based stereo algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(1):17–34, 1985.

[64] D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons, and A. Jain. A background model initialization algorithm for video surveillance. In *Int'l Conf. on Computer Vision*, pages 733–740, 2001.

[65] E. L. Hall, J. B. K. Tio, C. A. McPherson, and F. A. Sadjadi. Measuring curved surfaces for robot vision. *Computer*, 15(12):42–54, 1982.

[66] O. Hall-Holt and S. Rusinkiewicz. Stripe boundary codes for real-time structured-light range scanning of moving objects. In *Int'l Conf. on Computer Vision*, volume 2, pages 359–366, 2001.

[67] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: real-time surveillance of people and their activities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.

[68] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge University Press, 2000.

[69] M. Harville, G. Gordon, and J. Woodfill. Foreground segmentation using adaptive mixture models in color and depth. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 3–11, 2001.

[70] J. Heikkilä. Camera calibration toolbox for Matlab. `http://www.ee.oulu.fi/~jth/calibr/`, 2000.

[71] J. Heikkilä. Geometric camera calibration using circular control points. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, 2000.

[72] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(11):1168–1180, 1989.

[73] S. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representations. In *Int'l Conf. on Pattern Recognition*, pages 743–746, 1994.

[74] S. S. Intille and A. F. Bobick. Disparity-space images and large occlusion stereo. In *European Conf. on Computer Vision*, volume 2, pages 179–186, 1994.

[75] S. S. Intille and A. F. Bobick. Incorporating intensity edges in the recovery of occlusion regions. In *Int'l Conf. on Pattern Recognition*, pages 674–677, 1994.

[76] M. Isard and A. Blake. CONDENSATION—conditional density propagation for visual tracking. *Int'l Journal of Computer Vision*, 19(1):5–28, 1998.

[77] R. Jain and K. Wakimoto. Multiple perspective interactive video. In *Int'l Conf Multimedia Computing and Systems*, pages 202–211, 1995.

[78] A. D. Jepson and M. J. Black. Mixture models for optical flow computation. In *Computer Vision and Pattern Recognition*, pages 760–761, 1993.

[79] A. D. Jepson, D. J. Fleet, and M. J. Black. A layered motion representation with occlusion and compact spatial support. In *European Conf. on Computer Vision*, pages 692–706, 2002.

[80] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. In *Computer Vision and Pattern Recognition*, volume I, pages 415–422, 2001.

[81] D. G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *European Conf. on Computer Vision*, pages 395–410, 1992.

[82] G. A. Jones. *Stereo correspondence processes applied to linear image features.* PhD thesis, Kings College London, July 1993.

[83] G. A. Jones. Constraint, optimization, and hierarchy: reviewing stereoscopic correspondence of complex features. *Computer Vision and Image Understand*, 65(1):57–78, 1997.

[84] T. Kanade, A. Gruss, and L. Carley. A very fast VLSI rangefinder. In *Int'l Conf. on Robotics and Automation*, volume 39, pages 1322–1329, 1991.

[85] T. Kanade, K. Oda, A. Yoshida, M. Tanaka, and H. Kano. Video-rate Z keying: a new method for merging images. Technical Report CMU-RI-TR-95-38, Carnegie Mellon University, 1995.

[86] T. Kanade, P. W. Rander, and P. J. Narayanan. Virtualized reality: constructing virtual worlds from real scenes. *IEEE Multimedia, Immersive Telepresence*, 4(1):34–47, 1997.

[87] J. Kang, I. Cohen, and G. Medioni. Continuous tracking within and across camera streams. In *Computer Vision and Pattern Recognition*, volume 1, pages 267–272, 2003.

[88] K.-P. Karmann and A. von Brandt. Moving object recognition using an adaptive background memory. In *Proc. of 3rd Int'l Workshop on Time-Varying Image Processing and Moving Object Recognition*, pages 289–296, 1990.

[89] S. Khan, O. Javed, Z. Rasheed, and M. Shah. Human tracking in multiple cameras. In *Int'l Conf. on Computer Vision*, pages 331–336, 2001.

[90] V. Kolmogorov. Software. `http://www.adastral.ucl.ac.uk/~vladkolm/software.html`.

[91] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *Computer Vision and Pattern Recognition*, pages 407–414, 2005.

[92] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Int'l Conf. on Computer Vision*, pages 508–515, 2001.

[93] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conf. on Computer Vision*, pages 82–96, 2002.

[94] K. N. Kutulakos and S. M. Seitz. Theory of shape by space carving. *Int'l Journal of Computer Vision*, 38(3):199–218, 2000.

[95] J.-M. Lavest, M. Viala, and M. Dhome. Do we really need an accurate calibration pattern to achieve a reliable camera calibration? In *European Conf. on Computer Vision*, volume I, pages 158–174, 1998.

[96] T. Leung and J. Malik. Contour continuity in region-based image segmentation. In *European Conf. on Computer Vision*, volume 1, pages 544–559, 1998.

[97] R. Li and S. Sclaroff. Multi-scale 3D scene flow from binocular stereo sequences. In *IEEE Workshop on Motion and Video Computing*, page ??, 2005.

[98] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Computer Vision and Pattern Recognition*, pages 482–488, June 1998.

[99] M. H. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(8):1073–1078, 2004.

[100] H. Liu and P. Bhattacharya. Uncalibrated stereo matching using DWT. In *Int'l Conf. on Pattern Recognition*, pages 114–118, 2000.

[101] L. Liu and G. Fan. Combined key-frame extraction and object-based video segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(7):869–884, 2005.

[102] B. D. Lucas and T. Kanade. Iterative image registration technique with an application to stereo vision. In *Int'l Joint Conf. on Artificial Intelligence*, pages 674–679, 1981.

[103] J. Magarey and A. Dick. Multiresolution stereo image matching using complex wavelets. In *Int'l Conf. on Pattern Recognition*, pages 4–7, 1998.

[104] A.-R. Mansouri, A. Mitiche, and J. Konrad. Selective image diffusion: application to disparity estimation. In *Int'l Conf. on Image Processing*, volume 3, pages 284–288, 1998.

[105] S. B. Marapane and M. M. Trivedi. Multi-primitive hierarchical (MPH) stereo system. In *Computer Vision and Pattern Recognition*, pages 499–505, 1992.

[106] D. Marr and T. Poggio. A computational theory of human stereo vision. In *Proc. Royal Society of London*, volume B204, pages 301–328, 1979.

[107] A. Mittal and L. Davis. Unified multi-camera detection and tracking using region matching. In *Proc. 2001 IEEE Workshop on Multi-Object Tracking*, pages 3–10, 2001.

[108] J. More. *Numerical analysis*, chapter The Levenberg-Marquardt algorithm, implementation, and theory. Springer-Verlag, 1977.

[109] K. Muhlmann, D. Maier, J. Hesser, and R. Manner. Calculating dense disparity maps from color stereo images. In *Proc. IEEE Workshop Stereo and Multi-Baseline Vision*, pages 30–36, 2001.

[110] J. Mulligan, V. Isler, and K. Daniilidis. Trinocular stereo: a real-time algorithm and its evaluation. *Int'l Journal of Computer Vision*, 47(1/2/3):51–61, 2002.

[111] V. Nair and J. J. Clark. Automated visual surveillance using hidden Markov models. In *15th Vision Interface Conference*, pages 88–92, 2002.

[112] P. J. Narayanan, P. W. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Int'l Conf. on Computer Vision*, pages 3–10, 1998.

[113] R. M. Neal. *Bayesian learning for neural networks.* Springer-Verlag New York, 1996. Lecture Notes in Statistics, Vol. 118.

[114] O. Nestares and D. J. Fleet. Probabilistic tracking of motion boundaries with spatiotemporal predictions. In *Computer Vision and Pattern Recognition*, volume II, pages 358–365, 2001.

[115] B. North and A. Blake. Learning dynamical models using expectation-maximisation. In *Int'l Conf. on Computer Vision*, pages 384–389, 1998.

[116] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

[117] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Computer Graphics and Applications*, 22(6):64 – 71, 2002.

[118] N. M. Oliver, B. Rosario, and A. P. Pentland. A Bayesian computer vision system for modeling human interactions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[119] D. Ormoneit, H. Sidenbladh, M. J. Black, and T. Hastie. Learning and tracking cyclic human motion. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 894–900. The MIT Press, 2001.

[120] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. PMF: a stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1981.

[121] S. B. Pollard, J. Porrill, J. E. W. Mayhew, and J. P. Frisby. Matching geometrical descriptions in three-space. *Image and Vision Computing*, 5(2):73–78, 1987.

[122] E. Poon and D. J. Fleet. Hybrid Monte Carlo filtering: edge-based people tracking. In *IEEE Workshop on Motion and Video Computing*, pages 151–158, 2002.

[123] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara. Detecting moving shadows: algorithms and evaluation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(7):918–923, 2003.

[124] K. Prazdny. Detection of binocular disparities. *Biological Cybernetics*, 52(2):93–99, 1985.

[125] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[126] S. Randriamasy and A. Gagalowicz. Region based stereo matching oriented image processing. In *Computer Vision and Pattern Recognition*, pages 736–736, 1991.

[127] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH 98, Proc. 25th Annual Conf. Computer Graphics and Interactive Techniques*, pages 179–188, July 1998.

[128] M. Rioux, G. Bechthold, D. Taylor, and M. Duggan. Design of a large depth of view three-dimensional camera for robot vision. *Optical Engineering*, 26(12):1245–1250, 1987.

[129] E. M. Riseman and M. A. Arbib. Computational techniques in the visual segmentation of static scenes. *Computer Graphics and Image Processing*, 6(3):221–276, 1977.

[130] J. Rittscher, J. Kato, S. Joga, and A. Blake. A probabilistic background model for tracking. In *European Conf. on Computer Vision*, pages 336–350, 2000.

[131] S. Roy and I. J. Cox. A maximum-flow formulation of the N-camera stereo correspondence problem. In *Int'l Conf. on Computer Vision*, pages 492–499, 1998.

[132] J. Salvi, X. Armangué, and J. Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35:1617–1635, 2002.

[133] P. Sander, L. Vinet, and L. Cohen. Hierarchical region based stereo matching. In *Proc. 6th Scandinavian Conf. on Image Analysis*, pages 71–78, 1989.

[134] K. Sato and S. Inokuchi. Three-dimensional surface measurement by space encoding range imaging. *Journal of Robotic Systems*, 2(1):27–39, 1985.

[135] D. Scharstein. Matching images by computing their gradient fields. In *Int'l Conf. on Pattern Recognition*, volume 1, pages 572–575, 1994.

[136] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *Int'l Journal of Computer Vision*, 28(2):155–174, 1998.

[137] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int'l Journal of Computer Vision*, 47(1-3):7–42, 2002.

[138] C. Schmid and A. Zisserman. The geometry and matching of curves in multiple views. In *European Conf. on Computer Vision*, pages 104–118, 1998.

[139] J. Schmidt, H. Niemann, and S. Vogt. Dense disparity maps in real-time with an application to augmented reality. In *IEEE Workshop on Applications of Computer Vision*, pages 225–230, 2002.

[140] H. Schneiderman. Learning a restricted Bayesian network for object detection. In *Computer Vision and Pattern Recognition*, volume 2, pages 639–646, 2004.

[141] J. Shah. Nonlinear diffusion model for discontinuous disparity and half-occlusions in stereo. In *Computer Vision and Pattern Recognition*, pages 34–40, 1993.

[142] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *European Conf. on Computer Vision*, pages 702–718, 2000.

[143] C. C. Slama, editor. *Manual of photogrammetry.* Falls Church, Va.: American Society of Photogrammetry, 1980. 4th edition.

[144] X. Song and G. Fan. Joint key-frame extraction and object-based video segmentation. In *IEEE Workshop on Motion and Video Computing*, 2005.

[145] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[146] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[147] P. Sturm and S. Maybank. On plane-based camera calibration: a general algorithm, singularities, applications. In *Computer Vision and Pattern Recognition*, pages 432–437, June 1999.

[148] C. Sun. A fast stereo matching method. In *Proc. Digital Image Computing: Techniques and Applications*, pages 95–100, 1997.

[149] J. Sun, N.-N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[150] H. Tao, H. S. Sawhney, and R. Kumar. Dynamic layer representation with applications to tracking. In *Computer Vision and Pattern Recognition*, volume 2, pages 134–141, 2000.

[151] L. Taycher, J. W. F. III, and T. Darrell. Incorporating object tracking feedback into background maintenance framework. In *IEEE Workshop on Motion and Video Computing*, 2005.

[152] I. Thomo, S. Malasiotis, and M. G. Strintzis. Optimized block based disparity estimation in stereo systems using a maximum-flow approach. In *Proc. SIBGRAPI'98. Int'l Symposium on Computer Graphics, Image Processing, and Vision*, pages 410–417, 1998.

[153] C. Tomasi and R. Manduchi. Stereo matching as a nearest-neighbor problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(3):333–340, 1998.

[154] P. H. Torr, R. Szeliski, and P. Anandan. An integrated Bayesian approach to layer extraction from image sequences. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(3):297–303, 2001.

[155] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *Int'l Conf. on Computer Vision*, pages 255–261, 1999.

[156] B. Triggs. Autocalibration from planar scenes. In *European Conf. on Computer Vision*, pages 89–105, June 1998.

[157] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision.* Prentice-Hall, Inc., 1998.

[158] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J. Robotics and Automation*, 3(4):323–344, 1987.

[159] V. Venkateswar and R. Chellappa. Hierarchical stereo and motion correspondence using feature groupings. *Int'l Journal of Computer Vision*, 15(3):245–269, 1995.

[160] J. Y. Wang and E. H. Adelson. Layered representation for motion analysis. In *Computer Vision and Pattern Recognition*, pages 361–366, 1993.

[161] G. Wei and S. Ma. A complete two-plane camera calibration method and experimental comparisons. In *Int'l Conf. on Computer Vision*, pages 439–446, May 1993.

[162] Y. Weiss and E. H. Adelson. A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[163] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(10):965–980, 1992.

[164] R. Willson. Tsai camera calibration software. `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html`, 1995.

[165] C. R. Wren, A. J. Azarbayejani, T. J. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[166] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cut. In *Computer Vision and Pattern Recognition*, volume 2, pages 972–979, 2004.

[167] M. Xie and M. Thonnat. A cooperative algorithm for the matching of contour points and contour chains. In *Proc. 6th Int'l Conf. on Image Analysis and Processing*, pages 326–333, 1991.

[168] S. X. Yu and J. Shi. Object-specific figure-ground segregation. In *Computer Vision and Pattern Recognition*, pages 1–12, 2003.

[169] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *European Conf. on Computer Vision*, pages 150–158, 1994.

[170] L. Zhang, B. Curless, and S. M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *Int'l Symposium on 3D Data Processing Visualization and Transmission*, pages 24–36, 2002.

[171] Z. Zhang. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, http://research.microsoft.com/~zhang/ Calib/, 1998.

[172] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[173] Z. Zhang. Camera calibration with one-dimensional objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(7):892–899, 2004.

[174] T. Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *Computer Vision and Pattern Recognition*, pages 459–566, 2003.

[175] Y. Zhou, W. Xu, H. Tao, and Y. Gong. Background segmentation using spatial-temporal multi-resolution MRF. In *IEEE Workshop on Motion and Video Computing*, 2005.